1991年12月1日発行（毎月1回1日発行）第10巻12号通巻116号　昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

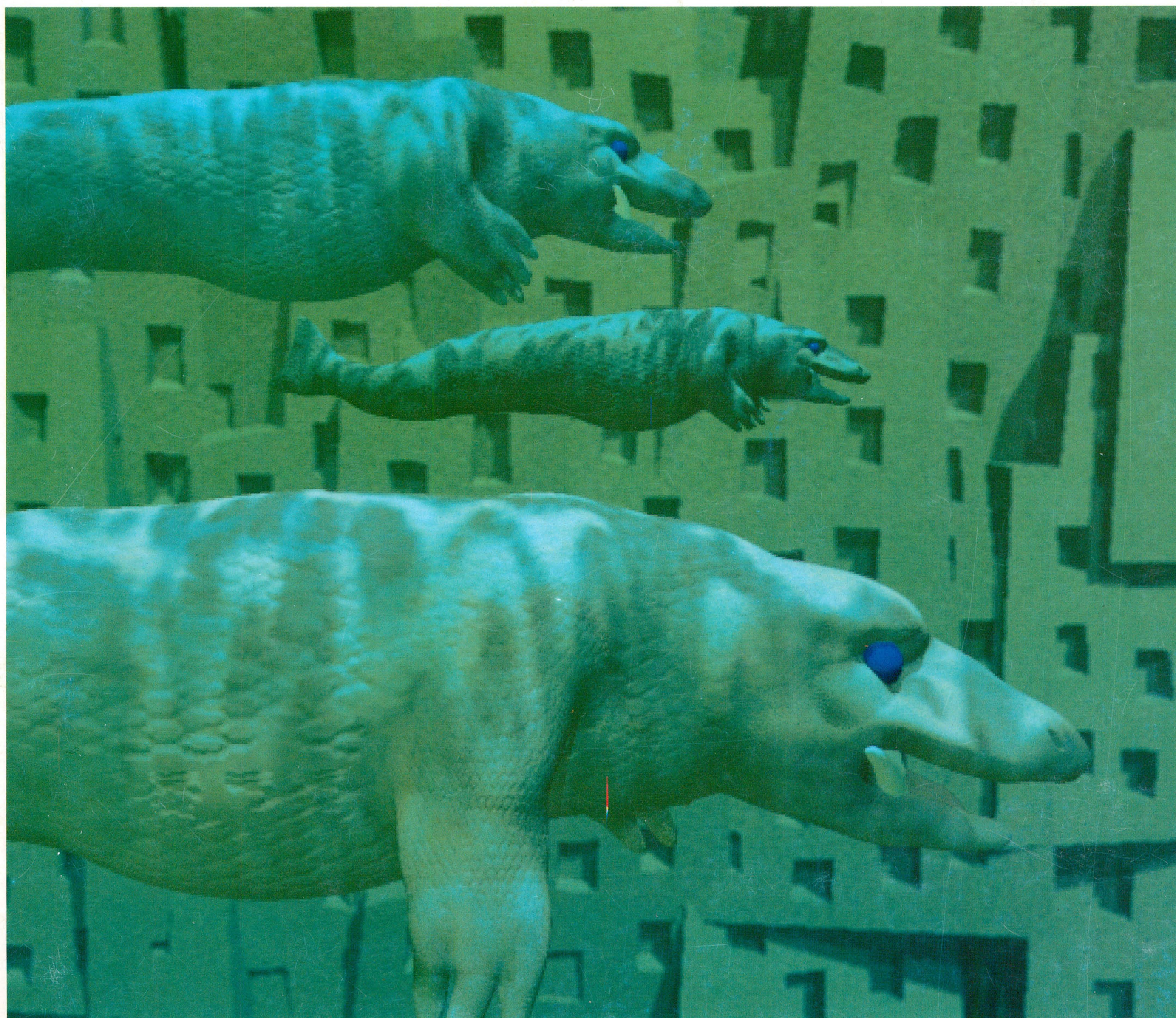
Oh!X

特集 音・そして音楽とコンピュータ

音色の合成と解析/Z-MUSICシステム&サンプルプログラム
エレクトロニクスショウ/データショウレポート
S-OS Small-C用SLANGコンパチライブラリ/V70ボード
別冊付録 X68000 THE GAME SOFTWARE BEST SELECTION

12
1991

SOFT BANK オー!エックス 定価600円



SHARP

アプリも使うけどオリジナルツールも創りたい。

X68000の世界に、思いきって踏み込んでみてください。アプリケーションの達人、ステーショナリーとしてのパソコン、それはそれで全く異論はないのですが、もっと新鮮な感動、驚き、発見に出会うはずです。コンピュータが本来持つ創造性、それとあなたの感性との接点に新しい何かが生まれる。グラフィック、サウンド&ミュージック、エンターテイメント、X68000はさまざまなフィールドで、あなたの才能に応えるクリエイティブ環境を備えています。

●クロック周波数16MHzの68000搭載 ●ウィンドウアプリケーションも続々登場、操作性を一段と高めたSX-WINDOW Ver.1.1搭載●メインメモリは標準で2MB、本体内に最大8MB、I/Oスロットを使えば最大12MBまで増設可能、数値演算プロセッサも本体内に取りつけ可能な高密度メモリ環境●大容量メディア対応、SCSIインターフェイス標準装備●X68000シリーズとフルコンパチブル設計。

瞬速16MHz、エクシヴィ快走。

X68000
PERSONAL WORKSTATION
XVI
エクシヴィ



本体+キーボード+マウス・トラックボール
CZ-634C-TN(チタンブラック) 標準価格368,000円(税別)
81MB HDタイプ CZ-644C-TN(チタンブラック) 標準価格518,000円(税別)

●写真はCZ-644C-TNと別売の15型カラーディスプレイテレビCZ-614D-TN標準価格135,000円(税別)

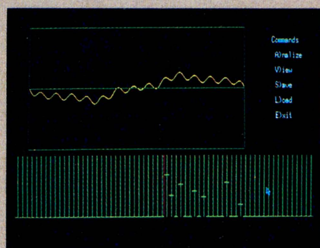
シャープX68000パソコン教室開催中
- 会場:四谷教室
- コース:入門コース・表集計コース・音楽コース・絵画コース
- 申込受付電話番号(03)3260-8365
- 受講料:2,000円(税別)

資料請求券 X68000 Oh!X 係

夢、創ります。「第1回全日本X68000芸術祭」作品募集中!

クリエイティブマインドを刺激する全国規模のビッグなオリジナルソフトウェア・作品コンテストです。
ゲーム、ミュージック、グラフィックなどの力作をぜひお寄せ下さい。詳細は店頭でポスター・チラシをご覧ください。

(九州地区予選大会〆切り迫る! 11月29日(金)必着)



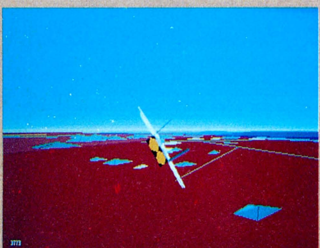
特集　音・そして音楽とコンピュータ



エレクトロニクスショウ＆データショウ



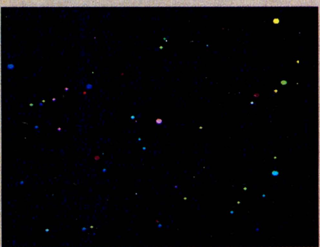
フェアリーランドストーリー



F-15ストライクイーグルII



大人のためのX68000



(て)のショートプロばーてぃ

Oh!X

C O N T

●特集

73 音・そして音楽とコンピュータ

74 音と音源を探る
音とはなにか　中野修一

76 FFT/逆FFTによる音声分析
冬の夜長のスペクトル解析　石上達也

81 AD PCMの超活用
FM音源の波形を創る　丹　明彦

90 MMLによる音楽表現法
Z-MUSIC公式ガイドブック　西川善司

93 Z-MUSIC LIVE SPECIAL SHINDO ON STAGE
ファイナルラップ2よりエンディング曲
ターボアウトランより KEEP YOUR HEART　進藤慶到

99 DTMへの招待
MIDIをめぐる環境　紀尾井誠

101 MIDIボードの使い方
MIDI出力方法論　牛島 健雄

●Oh!X 4周年特別企画

46 愛読者特別モニタ

●カラー紹介

28 エレクトロニクスショウ&データショウ

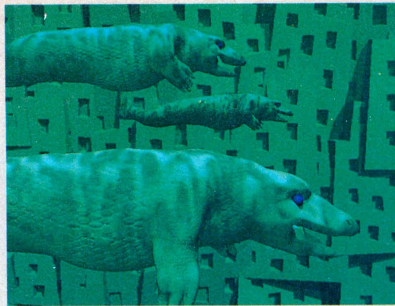
●THE SOFTOUCH

32 SOFTWARE INFORMATION
新作ソフトウェア/TOP10

GAME REVIEW
34 フェアリーランドストーリー　金子俊一
36 プロサッカー68　影山裕昭
38 機動戦士ガンダム クラシック・オペレーション　石上達也
40 ノーブルマインド　古村　聡
41 サイバーコア　高橋哲史
42 F-15ストライクイーグルII　西川善司

44 AFTER REVIEW
ボンバーマン

〈スタッフ〉
●編集長／前田　徹　●副編集長／植木章夫　●編集／岡崎栄子　浅井研二　山田純二　●協力／有田隆也　中森　章　林　一樹　荻窪　圭　華門真人　毛内俊行　吉田賢司　影山裕昭　古村　聡　村田敏幸　丹　明彦　三沢和彦　長沢淳博　宮島　靖　金子俊一　浦川博之　石上達也　●カメラ／杉山和美　●イラスト／永沢しげる　山田晴久　寺尾響子　●アートディレクター／島村勝頼　●レイアウト／元木昌子　AD GREEN　●校正／グループごじら



表紙絵：須藤　牧人

E　N　T　S

●シリーズ全機種共通システム

141 THE SENTINEL

142 Small-C用 SLANGコンパチ関数　伊藤直也

●読みもの

153 X-OVER NIGHT 第18話 BOOK　高原秀己

154 猫とコンピュータ 第65回 冷凍しちゃうぞ　高沢恭子

156 第54回 知能機械概論――お茶目な計算機たち―― 我々はぶざまな巨大ロボット?　有田隆也

●連載/紹介/講座/プログラム

30 響子inCGわ～るど［第7回］ プレゼント　寺尾響子

48 謎のV70ボードを追う V70とは何者か?　中野修一

50 ハードウェア工作入門〈18〉 ハイテクタンク製作（発展編）　三沢和彦

53 大人のためのX68000［第15回］ F-Card GTをいじくる　荻窪　圭

57 Oh!X LIVE in '91
OH YEAH!(X68000)　阿部俊光
サイレント・イヴ(X1/turbo)　佐々木孝司
おまけ　ジングルベル(X68000)　編集部

62 Creative Computer Music入門(3) メロディを生かす伴奏とは?　瀧　康史

66 ANOTHER CG WORLD　寺尾響子

68 吾輩はX68000である［第8回］ 愛のIOCSコール　泉　大介

111 X68000マシン語プログラミング Chapter_1D₁₁ 自由変形ルーチンの作成　村田敏幸

119 ようこそここへC言語［最終回］ ファイル入出力って何だろう　中森　章

133 マシン語カクテルin Z80's Bar 第27回 炎のプログラミング勝負　柴田　淳

149 (で)のショートプロぱーてぃ その27 ★よもう一度　古村　聡

Oh! X INDEX '91……158
ペンギン情報コーナー……162
FILES Oh! X……164
Oh! X 質問箱……166
編集室から/DRIVE ON/ごめんなさいのコーナー/SHIFT BREAK/microOdyssey……172

1991 DEC.
12

■広告目次

アイビット電子……180
アクセス……184
R&Rメディア……178
OAシステムプラザ……177
オーエーブレイン……182
オーエーランド……16
計測技研……181
コナミ……10・11
サザンエンタープライス……183(下)
サン・ミュージカル・サービス 183(上)
J&P……表3
シャープ……表2・表4・1・4-8
九十九電機……23
デンキヤ……179
野邊ゲームデザイナーズアカデミー……176
パソコンプラザオクト……18・19
ビクター音楽産業……15・17
P&A……20・21
ブラザー工業……12・13
満開製作所……175
メディックス……14
ライトスタッフ……9
ワールドインアオヤマ……22

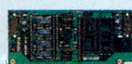
お望みのパワーシステムへ。

シャープペリフェラルファミリー
X68000

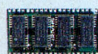
PRO II

ボード

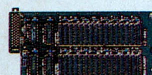
拡張メモリ



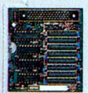
2MB増設RAMボード
(CZ-634C/644C専用)
CZ-6BE2A
標準価格59,800円(税別)
※2MB増設RAM(CZ-6BE2B)専用ソケットを2個用意しています。



2MB増設RAM
(CZ-634C/644C専用)
CZ-6BE2B
標準価格54,800円(税別)
※本増設RAM(CZ-6BE2B)は、2MB増設RAMボードが必要です。CZ-6BE2A上の専用ソケット(2個用意)に装着ください。
※取付に関してはシャープお客様ご相談窓口にてご相談ください。



1MB増設RAMボード
(CZ-600C専用)
★CZ-6BE1
標準価格35,000円(税別)



1MB増設RAMボード
(CZ-601C/611C/652C/653C/662C/663C用)
CZ-6BE1B
標準価格28,000円(税別)

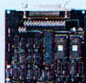


2MB増設RAMボード※6
CZ-6BE2
標準価格79,800円(税別)

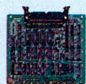


4MB増設RAMボード※6
CZ-6BE4
標準価格138,000円(税別)

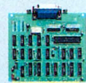
インターフェイス



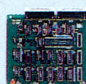
SCSIボード※7
CZ-6BS1
標準価格29,800円(税別)
(ソフトウェア(SCSIユーティリティ)同梱)



ユニバーサルI/Oボード
★CZ-6BU1
標準価格39,800円(税別)

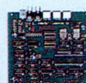


GP-IBボード
★CZ-6BG1
標準価格59,800円(税別)



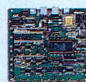
増設用RS-232Cボード
(2チャンネル)
★CZ-6BF1
標準価格49,800円(税別)

MIDI



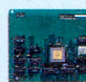
MIDIボード
CZ-6BM1A
標準価格26,800円(税別)

FAX



FAXボード
CZ-6BC1
標準価格79,800円(税別)

数値演算プロセッサ



数値演算プロセッサボード
CZ-6BP1
標準価格79,800円(税別)

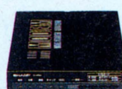


数値演算プロセッサ
(CZ-634C/644C専用)
CZ-6BP2
標準価格45,800円(税別)
※取付に関してはシャープお客様ご相談窓口にてご相談ください。
※特別ケース入りです。



ネットワーク

モデム



モデムユニット※8
CZ-8TM2
標準価格49,800円(税別)
(RS-232Cケーブル同梱)

RS-232Cケーブル

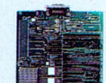


RS-232Cケーブル
(平行接続型)
CZ-8LM1
標準価格7,200円(税別)



RS-232Cケーブル
(クロス接続型)
CZ-8LM2
標準価格7,200円(税別)

LANボード



LANボード
CZ-6BL1
標準価格268,000円(税別)
(イーサネット用)

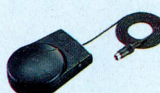


CZ-6BL2
標準価格298,000円(税別)
(イーサネット/チーパネット両用)
※電源ユニット・ソフトウェア
(ネットワークドライバVer1.0)同梱

入力



インテリジェントコントローラ
CZ-8NJ2
標準価格23,800円(税別)



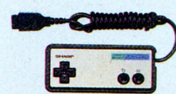
マウス・トラックボール
CZ-8NM3
標準価格9,800円(税別)



トラックボール
CZ-8NT1
標準価格13,800円(税別)



マウス
CZ-8NM2A
標準価格6,800円(税別)



ジョイカード
CZ-8NJ1
標準価格1,700円(税別)

その他

拡張スロット



拡張I/Oボックス(4スロット)
(CZ-600C/601C/602C/603C/604C/611C/612C/613C/623C/634C/644C用)
CZ-6EB1-BK
★CZ-6EB1
標準価格88,000円(税別)

スピーカー



アンプ内蔵
スピーカーシステム(2本1組)
AN-S100
標準価格36,600円(税別)

システムラック



システムラック
(CZ-600C/601C/602C/603C/604C/611C/612C/613C/623C/634C/644C用)
CZ-6SD1
標準価格44,800円(税別)

■本広告に掲載しております拡張ボード類のうち、CZ-634C/644Cの16MHzモードで動作しないものが一部あります。 ★印の商品は在庫僅少です。
■製品改良のため仕様の一部を予告なく変更することがあります。またこの広告の色調は印刷のため実物とは多少異なる場合もありますのであらかじめご了承ください。

CZ-600C用)、CZ-6BE1B 標準価格28,000円(税別・CZ-601C、CZ-611C、652C、653C、662C、663C用)を増設してください。 ※7 CZ-600C、601C、602C、603C、611C、612C、613Cに装着の場合、I/Oスロット2に装着ください。CZ-652C、653C、662C、663Cに装着の場合はI/Oスロット4に装着ください。また、CZ-6BG1、6BU1、6BL1、6BL2、6BN1などのボードは、接続コネクタとの関係で本ボードとの併用はできませんのでご注意ください。なお、本ボードはX68000用OS Human 68K ver.2.0以上にてご使用ください。 ※8 モデムユニットCZ-8TM2に同梱のソフトはX1/X1ターボシリーズ用です。

電子機器事業本部 AVCシステム事業推進室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)

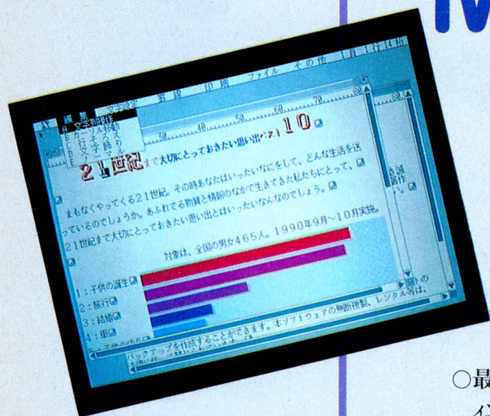
SHARP

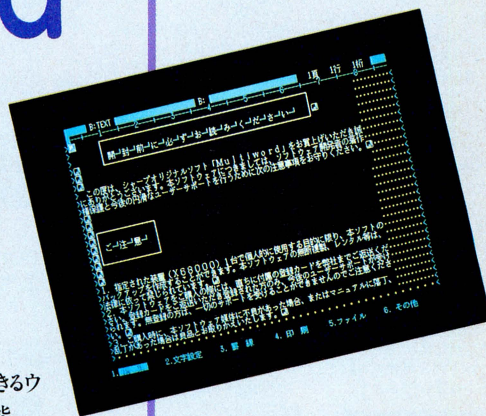
多彩なグラフィック機能搭載
多機能ワープロ

マルチワープロ PRO-68K

Multiword

CZ-225BS　標準価格32,000円(税別)

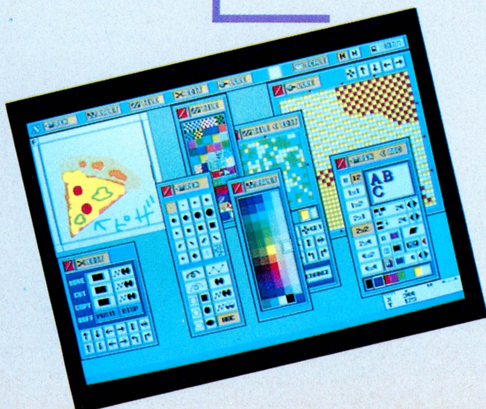




WYSIWYGを採用したウィンドウモード、エディタ感覚で入力できるテキストモード。さらにクリエイティブマインドを刺激する多彩なグラフィック機能を搭載。X68000のパフォーマンスをフルに活かした、ヒューマンなワープロの誕生です。

○最大10文書までの複数文書を同一画面で編集できるウィンドウモード装備。文書間でのカット&ペーストも可能
○スピーディな文字入力をサポートするテキストモード
○20種類のペン先を使って自由にグラフィックを作成できるグラフィックエディタを装備。タイルパターンは52種類、オリジナルパターンも作成可能
○影付文字、袋文字、斜体文字など多彩な文字種、文字間隔もドット単位に指定可能
○ビジネス文書に威力を発揮する豊富な改行・罫線機能
○用途に合わせて選べる幅広いプリンタサポート。多彩な用紙設定、印刷設定で思い通りのアウトプットが可能
○イメージスキャナ入力は、パラレルインターフェイスに対応。ハンディスキャナ入力もサポート。

※メインメモリ2MB必要です。



「Multiword」発売記念キャンペーン実施中!!

「Multiword」CZ-225BSの発売を記念し、期間中にご購入の方にステキな賞品が当ります。この機会にぜひご購入ください。

● 期間：平成3年8.1～12.31迄(消印有効)
● 対象：「Multiword」ご購入ユーザーで登録カードを弊社に送付された方の中から、厳選な抽選により決定いたします。
● 発表：パソコン専門誌X68000ソフト広告誌上で発表します。
● 賞品：1等/X68000フロッピーアタッシュケース……5名
2等/X68000シースルークロック……10名
3等/X68000キーホルダー/ネクタイピン……15名
4等/X68000マイウェイバッグ……20名
5等/X68000特製テレフォンカード/ステッカー……30名

資料請求券
CZソフト
Oh!X
12係

SHARP

めざせ!グランプリ パソコンオリジナル作品コンテスト

「夢、創ります。山下章氏プロデュース」

第1回全日本X68000 芸術祭



X68000アイドル山下章氏司会、進行による
ユーザー参加型作品コンテスト

■主催：シャープ株式会社 電子機器事業本部 システム機器営業部
■共催：シャープエレクトロニクス販売株式会社各統轄営業部
東京中央シャープ販売㈱・浪速シャープ電機㈱・沖縄シャープ電機㈱
■協賛：出版社・ソフトハウス・サードパーティ・主要販売店

作品大募集中!!

パソコンファンなら全員参加。
なんでもアリの作品コンテスト!
個性が光る作品、ドンドン応募して下さい。
地区大会を勝ち抜いて、
夢は全国大会グランプリ!

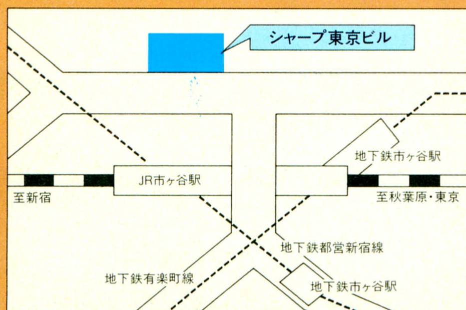
当日は会場へ大集合!!

会場に来たみんなが審査員に。
「山下章の裏ワザ講座」「MIDIライブ」
も迫力満点!
X68000リファレンスBookもプレゼント。
たくさん友達を誘って参加して下さい。

首都圏地区大会(東京)

11月24日(日) 13:00〜16:00

●会場/シャープ東京支社 8Fエルムホール●対象都道府県/埼玉・山梨・千葉・新潟・東京●問い合わせ先/〒162 東京都新宿区市谷八幡町8 シャープエレクトロニクス販売㈱首都圏統轄(営)パソコン営業部 ☎03-3266-8248

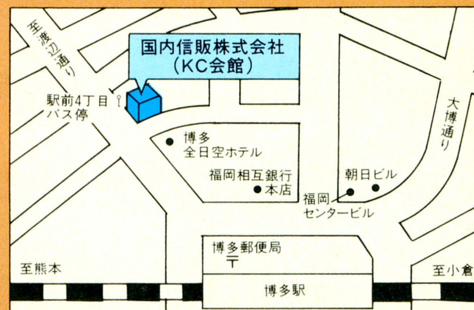


九州地区大会(福岡)

12月14日(土) 14:00〜17:00

■応募締切り間近!!(11月29日(金)必着)

●会場/KC会館 2F大ホール●対象都道府県/福岡・佐賀・長崎・熊本・大分・宮崎・鹿児島・沖縄●応募・問い合わせ先/〒816 福岡市博多区井相田2-12-1 シャープエレクトロニクス販売㈱九州統轄(営)パソコン営業部 ☎092-501-6806



【作品応募要項】：平成3年7月改訂

◆作品基準：パーソナルコンピュータ(メーカー、機種を問わず)で制作した、オリジナル未発表のプログラム、グラフィックス、コンピュータ・ミュージック等であること。なお、応募者はシャープに対し、応募作品を自由に利用する独占的権利を無償にて許諾するものとします。また、応募作品は返却致しませんので、コピーをとってからご応募下さい。**◆部門**：①ゲーム部門②ミュージック部門(自作の曲/一般曲・ゲームミュージックのアレンジ等、MIDI使用も可。)③グラフィックス部門(Z's STAFF PRO-68K,DOGA等のツールを使用して描いたものなど画面上に表示されるグラフィックスなら何でも可。)④その他部門：ユーティリティ/一発ギャグ/パフォーマンス/ビジネス利用/その他)＊応募は、1部門につき1人1作。1人複数部門応募は可。又団体制作も可。**◆応募資格**：各地区大会の対象都道府県在住の方。補選は全国より各地区大会未応募の方。**◆応募方法**：フロッピー・ディスクでご応募下さい。(グラフィック部門は、ビデオテープでの応募も可。但し、コンピュータ用の自作ソフトであることを証明する為に、必ずプログラムディスクを添えて送って下さい。)住所/氏名/年齢/職業(学校名・学年)/電話番号/開発に要した期間/開発に使用・利用したツール名/セールスポイント/取り扱い上の注意/動作に必要とする特殊機材を明記した用紙を添え、各地区の応募先まで郵送して下さい。締め切りはその地区の地区大会開催日の2週間前(必着)です。**◆審査員**：一般来場者・特別審査員各位**◆賞・賞品**：《地区大会》◇大賞(1点)トロフィー、賞状、副賞：5万円相当のシャープ製品、全国大会へのエントリー権◇入選(首都圏3点、近畿2点、中部・九州各1点、他地区なし)賞状、副賞：3万円相当のシャープ製品、全国大会へのエントリー権◇参加賞(大賞・入賞以外)X68000オリジナルグッズ◇協賛各社賞 《全国大会》◇第1回全日本X68000芸術祭グランプリ(1点)トロフィー、賞状、副賞：「光磁気ディスクユニット(CZ-6MO1)」及び「ペアでの海外旅行(旅行クーポン50万円分)」(地区大会副賞を含め、総額100万円相当)◇各部門賞(各1点、計4点)賞状、副賞：30万円相当のシャープ製品◇協賛各社賞

※詳細は店頭のチラシをご覧下さい。

●各地区大会に応募載けなかった方には、平成4年2月に補選を予定

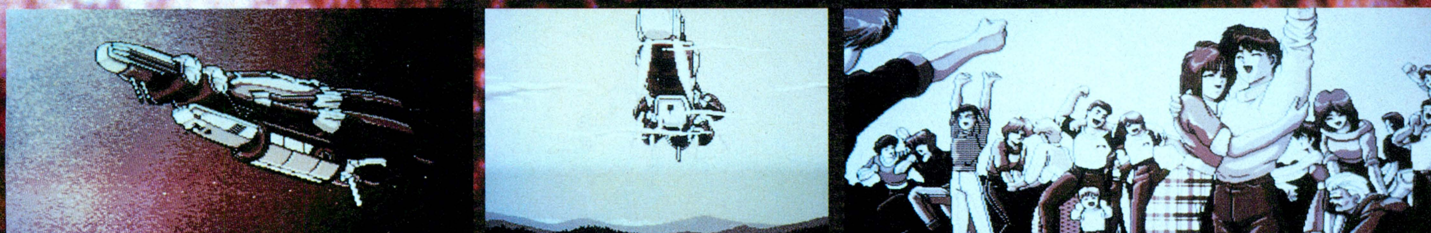
●全国大会は平成4年4月東京にて開催予定

協賛社(敬称略、順不同)：I/O、LOGIN、Oh!X、POPCOM、アスキー、コンプティーク、マイコン、マイコンBASICマガジン、DEMPAマイコンソフト、SPS、T&Eソフト、アイレム、ウルフチーム、エニックス、コナミ、システムサコム、システムソフト、ズーム、ダットジャパン、ハドソン、ホームデータ、マイクロキャビン、マイコンソフト、リバーヒルソフト、光栄、ローランド、エジソン、AVCフタバ電機、CBK、ECCSマルゼンムセン、ICカプセル、ICワールドトツカ、JCN、MZイン松原、OAアプリケーションズ、OAシステムシャープ、OAシステムプラザ、OAショップアクソン、OAナガシマ、OAランド、TNKソフト、Tゾーン、YET、アイ・ピー・エル、アイビット電子、アダチコンピュータ館、アプライド、いわきマイコンショップ、ウェーブ・アイ、エイコー丸亀店、エイトシステム、エイトピア、オービック、オガワムセン、オノデン、カインドソフト、カクタ、カトー無線電機、カホ無線、かわいソフト企画、グッドウィル、コスモス、コナン販売、コマツパソコンセンター、コムイン、コムライン、コムロード、コンパス、コンピュータバンク、サイアイ無線、サトームセン、サンミュージック、システムイン吉野、システムハウスラム、シスペック、シマコーシステム、ショーエイ、シントク、ジャスコ、ジャルク、スイテック、すみや、セイデンマツフジ、セキド、そうご電器、ソフトハウスポップ、タケベ無線、ダイイチデオニー、ダイイチパソコンCity、ダイエー、ダイオーICコスモランド、ダイデンアクセス店、だるまや西武、てくのらいふさわだ、デジタルシステム、デンコードー、トキハ、トロン、ナカウラ、ニイデンキ、ニチイ、ニノミヤ、ノジマ、ハイランド、ハドソン、バイトイン、バップス、パソコンショップキャル、パソコンランド21、ヒロセムセン、ビレイ、ピー・アンド・エー、ピック、フロッピア、ベストマイコン、ベスト電器、マイコンセンターウエノ、マイコンセンターツギタ、マイコンテック、マイコンハウス、マイコンランド上田、マイパソコンショップ五条、マツヤデンキ、マルツ電波、ミオス、ミナミ無線電機、ムーンベース、ムラウチ、メディアネットワーク西日本、メディア旭川、メルバ、ヤナゲン岐大ホームセンター、ヤマギワ、ラオックス、ランダム、リーダーズプロ、リードコナン、ロケット、ロジック、ワールドインアオヤマ、井上松影堂、鳥城無線、栄電社、億人、河合無線、関彰商事、丸栄でんき、丸善ムセン電機、喜多電機商会、岐阜コンピュータサービス、岐阜マイコンセンター、九十九電機、計測技研、光洋無線電機、三共ジョーシンJ&P、酒井電化センター、庄子デンキ、松本無線パーツ、上新電機J&P、西武パソコンプラザ、石丸電気マイコンセンター、石見電業社、大学生協、大竹商会、大洋無線、第一家電、第一無線工業、中京マイコン、電化センター、電巧堂チェーン、日本インコム・テレニック、日本インコム・ニック、日本マイコン流通センター、日本電子システム販売、日野電気、馬場電機、浜松マイコンセンター、宝谷楽器、豊栄家電、野田屋電機

●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表) シャープ株式会社

RIGHT STUFF

虚空に奔(はし)る創星のきらめきに遥かなる銀河の声(メッセージ)を聞く。



ALSHARK

X68000版　11月29日発売!!

惑星フィールドと宇宙空間を完全に別次元として捉え、「今一つ満たされない」という既存のスペースものの常識を打ち破ることに成功!!

各惑星に散らばる町は、30を数え、内部マップ100以上、会話データ20万字と、従来のRPGを凌駕するデータ量。

フィールドにおける戦闘モードは、同一マップ上でも地形によって出現するモンスターと背景が変化し、その攻撃パターンのグラフィックによる演出の数々は、見事というほかない凝りよう。さらに、宇宙における戦闘は、ファイティングキャリアー"アトライア"の性能を余すところなく再現!!

7つの恒星系とそれに付随する24の惑星上で繰り広げられるドラマの数々に、君は壮大な物語の主人公となりきることができるか。





9,800円(税別)

■MIDI対応　■5"2HD(5枚組)

PC9801シリーズ(VM、UV以降)
好評発売中!!　定価9,800円(税別)

- 5"2HD、3.5"2HD(各5枚組)
- 要640KB
- サウンドボード対応
- ジョイスティック対応
- 16色専用
- 要400ラインCRT(アナログRGB端子付き)



RS　株式会社 ライト スタッフ
〒140 東京都品川区西大井6-10-10 品川RSビル TEL.03-3772-5131
ユーザーテレフォン:03-3772-5073(ライトスタッフの最新情報をお知らせしています)

全国のパソコンショップ・デパートでお求め下さい。通信販売をご希望の場合は、現金書留、郵便振替(東京4-6099)で、商品名、機種、メディア、及び住所、氏名、電話番号を明記の上、弊社までお申し込み下さい。＊表示価格には消費税は含まれておりませんので、価格の3%の消費税を添えてお申し込み下さい。

もうすぐロクハチツインビーに会える!!

いよいよ出るぞ!!
楽しさいっぱい 出たなツインビー








ハネを撃つとくるくるまわる風車野郎。
コイツの弱点は赤く光るコアだ。
うまくまわして狙い撃て!


行く手をさえぎるお邪魔な敵には、
パワーをためてビッグショット攻撃だ。


ウネウネ動く、ブキミな背景の雲。
酔わないように気をつけろ!

X 68000「出たな!! ツインビー」は、ローランド社[MT-32][CM-32L]及び新音源[SC-55]に対応しています。

Roland

MIDI
MIDI対応機種


No Copy
このマークは
不法コピー
禁止マークです
ソフトウエア法的保護監視機構

札幌営業所　〒060 北海道札幌市中央区北1条西5-2-9　　福岡営業所　〒810 福岡県福岡市中央区天神2-8-30

ブラザー工業株式会社
〒467 名古屋市瑞穂区苗代町2番1号
TAKERU事務局 (052)824-2493
東京営業所(03)3274-6916
大阪営業所(06)252-4234

パソコンソフト自販機
TAKERU

HEAVY NOVA

熱い興奮!!
君は宇宙最強戦士
『Heavy Nova』(ヘビー ノヴァ)
になれるか!?

地球圏防衛軍の中核を為す部隊
「Heavy Dool」(ヘビー ドール)隊。

そのパイロット養成所から今、壮大な物語が始まろうとしている

目的はただ1つ!
宇宙で偉大な戦士だけに与えられる称号
『Heavy Nova』を獲得することだ!
君はHeavy Doolを操り、
果たして宇宙最強の戦士になれるか?

友達とも対戦できる!(2人プレイ可能)

12月20日発売予定
¥5,800(税込)
■対応機種:X68000
■企画/開発:株式会社マイクロネット

X68000初の低価格カード型データベース
X68000版 好評発売中

カード型DB	ツリー型メニューをはじめ初心者にも使いやすい設計。入力した数字の読み合わせ機能も付いて入力ミス無し!
リレーショナルDB	ウインドウ形式で表示される関係ファイルからの引用や加算減算が可能。HELP機能付きで簡単操作。
テキストDB	カードに記入したテキストファイルを表示させたり、ファイル内容の検索が可能です。
プログラムDB	カードにはデータの他にプログラム名が記入でき、起動もできますので、メニューソフトとしても使えます。
グラフDB	グラフファイル名をカードに記入しておけば、必要なデータと一緒にグラフもデータベース化できます。
画像DB	グラフィックソフトやスキャナで作成した画像ベタファイル名をカードに記入しておけば画像も表示可能です。



▲ツリー型メニュー
メニューを表示しただけで、F-Cardの持つ機能がすべて表示されます。ソフトをはじめて使う人や、たまにしか使わない人でも、迷わず必要な機能が引き出せます。

■企画/開発 クレスト
TAKERU価格……¥8,000(税込)

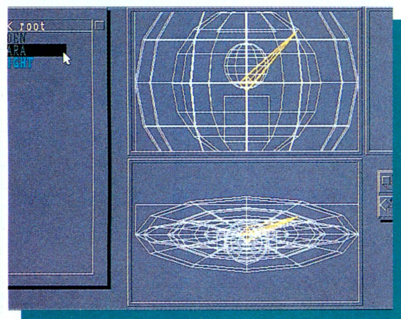
…電機 (0552)32-5033●長野 ダイエー長野店7F (0262)27-1311/ラオックス ヒナタ コンピューター館 (0262)37-2221●上田 マイコンランド西友上田店 (0268)26-3969●松本 遠兵 (0263)32-6350●金沢 うつのみや片町店 (0762)21-6136●富山 丸…カラー駅前店 (0764)41-2500●福井 エジソン (0776)26-2228/だるまや西武7F マイコンショップ (0776)27-0111●沼津 メルバ沼津 (0559)22-4858/すみやパソコンアイランド沼津店 (0559)23-5700●静岡 メルバ静岡 (054)254-5338/すみやパソコン…ランド静岡国吉田店 (054)263-5900/すみやパソコンアイランド静岡駅前店 (054)255-8819●富士 メルバ富士店 (0545)51-8022●焼津 メルバ焼津店 (054)626-0181●浜松 メルバ浜松本店 (053)464-8412/ホーエー家電有楽店 (053)453-1441●浜松…ロード浜松店 (053)454-0615●名古屋 EIDENテクノ名古屋 (052)581-1241/パソコンショップコムロード (052)263-5828/ちくさ正文館書店ターミナル店 (052)732-3601/カトー無線電機電気館4F (052)264-1534/マルゼン無線 第一アメ横店 (052)263…26/EIDENテクノ大須 (052)252-0181/トップカメラ4F (052)971-0111/J&P大須店 (052)262-1141/EIDENメディア大須 (052)242-8591/マルゼンムセン名古屋第2アメ横店 (052)263-6162●小牧 EIDENテクノ小牧 (0568)75-4261●名古屋…DEN鳴海店 (052)895-2271●豊橋 EIDENテクノ豊橋 (0532)52-1231●岡崎 ジャスコ岡崎店3F (0564)23-4960/EIDEN岡崎店 (0564)53-2722●豊田 ジャスコ豊田店パソコンショップJPC (0565)35-1761●岐阜 パソコンショップコムロード岐阜 (0582)66-0288●大垣 スイテック大垣ヤナゲン店A館6F (0584)81-3491●津 KawaiOA 津店 (0592)26-0111●四日市 Kawaiカデンワールド四日市 (0593)54-3366●伊勢 河合ムセンELFA 3F (0596)22-1111●松阪 KawaiBax店6F…98)26-0111●大津 西武百貨店大津 (0775)25-0111●日本橋 J&Pテクノランド (06)634-1211/J&Pメディアランド (06)634-1511/ニノミヤエレランド (06)632-2038/ニノックスコア日本橋店 (06)647-2038●難波 ニノミヤパソコンランド難波店 (06)643…17/J&Pコスモランド (06)634-3111●梅田 ニノックス駅前第4ビル店 (06)341-2031/エキサイト阪急三番街店 (06)374-3311●高槻 J&P高槻店 (0726)85-1212●八尾 西武百貨店八尾店 (0729)97-0111●京都 J&P京都近鉄店 (075)341-5769/…ヤマムセンやましな電遊館 大翔第一ビル1F (075)595-0200●和歌山 J&P和歌山店 (0734)28-1441●神戸 星電社三宮本店C-SPACE (078)391-8171/ダイエーさんのみや電器館パレックス (078)391-7911/J&Pさんのみや1ばん館 (078)231-…1●姫路 J&P姫路店 (0792)22-1221/姫路コンパックス (0792)94-8244●岡山 岡山VIVRE21 (0862)32-8881/ダイイチ岡山パソコンCiTY (0862)27-3011/ベスト電器岡山OA館(ジョリービル) (0862)23-7107●倉敷 ファソランド (0864)25-4701●…島 松本無線パーツ (082)243-4451/ダイイチ広島パソコンCiTY (082)248-4343/ダイイチ五日市パソコンCiTY (0829)24-2111●呉 ダイイチ呉パソコンCiTY (0823)25-6511●高松 Sound Check MOVECO.,LTD. (0878)61-6171●徳島…そごう7F (0886)25-4056●高知 デンキのタグチ (0888)23-0181●松山 ダイイチ松山パソコンCiTY (0899)31-6711●北九州 ベストマイコン小倉パソコン館 (093)551-6281●福岡 ベストマイコン福岡店 (092)781-7131/寿屋エレデ博多 (092)281…11●北九州 黒崎パソコンステーション (093)621-3541●久留米 ベスト電器久留米OA館 (0942)38-0111●長崎 ベスト電器長崎パソコン館 (0958)28-1333●佐世保 ベスト電器佐世保OA館 (0956)22-8660●熊本 寿屋本荘店 (096)372-5411/ベス…マイコン熊本パソコン館 (096)332-4180/J&P熊本店(ファインビルB1F) (096)359-7800●鹿児島 ベスト電器鹿児島パソコン館 (0992)23-2081●大分 ベストマイコン大分パソコン館 (0975)32-9396●宮崎 宮崎寿屋百貨店7F (0985)27-4111●宮崎 ベ…電器宮崎パソコン館 (0985)22-8325●那覇 ベスト電器那覇店6F (0988)62-7588●山形 デンコードー山形本店 (0236)23-1055●酒田 デンコードー酒田店 (0234)24-5885●一関 デンコードー一関店 (0191)25-2440●福島 オリエンタルレコー…0245)21-2101●池袋 ソフトピア池袋店 (03)3985-3268●横浜 ユニオン横浜店 (045)611-1543●藤沢 フォーミュラ (0466)24-3923●岩国 ダイイチ岩国店 (0827)21-2111●徳山 ダイイチ徳山店 (0834)21-1590●鳥取 ダイイチ鳥取店 (0857)…2222

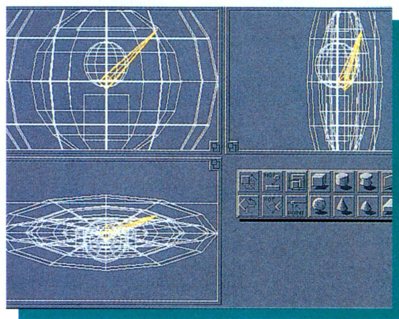
通信販売 通信販売をご希望の方は、ソフト名・機種名・住所・氏名・電話番号を明記の上TAKERU事務局まで現金書留でお申し込み下さい。
代金引換は一度、現金書留で申し込んで頂いた方に案内させて頂きます。

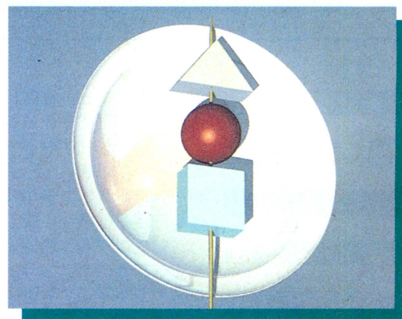
レイトレは、ここまで使える
フルマウスオペレーション
つみき感覚なモデリング
もっと欲しい色がある
絵心なんかいらない
西武線はえらい
町内で一番...

だから何なの?









なぜか今までありそうでなかったフルマウスオペレーションモデラー標準装備のレイトレーシングツール
なぜか今までありそうでなかったX68000 にだけ許された1024×512 画面モードを駆使した快適な操作環境...
なぜか今までありそうでなかった絵心がなくてもおでんが簡単に作れる簡易ポリゴンエディタ標準装備

つるつるぴかぴかのビー玉一個を夢見ていた人から座標計算のモデリングに疲れたプロの方に.....

MIRAGE（ミラージュ） MODEL（モデル） STUFF（スタッフ）

X68000対応版　近日発売

定価29,800円（税別価格）

MEDIX inc.
MEDIa miX Explorer
Head Office TOKYO JAPAN.
CG Design & P.C.B CAD
株式会社メディックス
〒165 東京都中野区江原町2-22-4 TEL.03-3950-2222

情報満載MMネット
24時間 開局中！
2400MNP/4,N81
ID:GUEST
Pass Word:GUEST
アクセスポイント
東京03-3950-1507

STAR WARS®

Attack on the DEATH STAR

スター★ウォーズ

アタック・オン・ザ・デス・スター

驚異の技術力による超高速3D処理の爽快なスピード感
映画で使われた音声と効果音のサンプリングでの再現による興奮の臨場感
自分のプレイを再現できるトレース機能

X68000版 11月22日発売予定 ¥7,200（税抜）

企画開発：M. N. M *Software*

発売：ビクター音楽産業株式会社

通信販売 当社の商品をお近くのパソコンショップでお買い求めになれない場合、商品名、機種名、住所、氏名、電話番号を明記のうえ、下記住所まで定価プラス3%消費税分を現金書留にてお申し込み下さい（送料無料）〒151 東京都渋谷区千駄ヶ谷2-8-16 ビクター音楽産業㈱（通信販売係）

全国通販

SHARP 認定 PPO-SHOP **O.A.ランド**

(TEL) 03-3770-8855

■アフターサービス万全のサポート体制

●下取・買取は電話で見積りしております。責任を持って下取りさせて頂きます。

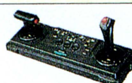
営業時間
平日……… AM10:00～PM7:00
土日・祭日… AM10:00～PM6:00

▶11・18～12・17

SHARPのことならなんでもおまかせ!!

大徳買セール!安く値切ってネ。(本体セット:送料 消費税込み)
お電話下さい。㊙価格をお知らせいたします。

流通事情により、広告表示価格は、お安くなる場合がありますので、ドンドンお電話下さい。



CYBER STICK
■CZ-8NJ2 (定価￥23,800)
OAランド特価 ▶￥18,000



電子手帳 ●見やすい漢字4桁表示!! 情報任時代の必需品!!
■PA-9500 (￥48,000)…▶特価￥38,000
■PA-8500 (￥28,000)…▶特価￥15,000
■PA-7500 (￥22,000)…▶特価￥12,000

低金利クレジットをご利用下さい。平日AM10時～PM7時、土日・祭日AM10時～PM6時迄ガンバッテま～す!!

SHARP X68000シリーズセット(送料・消費税込み)

X68000XVI

①CZ-634-TN+CZ-614D-TN
定価合計￥503,000

12回	￥32,800
24回	￥17,400
36回	￥12,100
48回	￥ 9,500



②CZ-634C-TN+CZ-607D-TN
定価合計￥467,800

12回	￥30,700
24回	￥16,300
36回	￥11,300
48回	￥ 8,900

③CZ-634C-TN+CZ-606D-TN
定価合計￥447,800

12回	￥29,200
24回	￥15,500
36回	￥10,800
48回	￥ 8,400

■CZ-634C■ 特価 ￥TEL下さい!!

■CZ-644C■ 特価 ￥TEL下さい!!

X68000XVI-HD

①CZ-644C-TN+CZ-614D-TN
定価合計￥653,000

12回	￥42,600
24回	￥22,600
36回	￥15,700
48回	￥12,300

②CZ-644C-TN+CZ-607D-TN
定価合計￥618,700

12回	￥40,600
24回	￥21,400
36回	￥14,900
48回	￥11,700

③CZ-644C-TN+CZ-606D-TN
定価合計￥597,800

12回	￥39,000
24回	￥20,700
36回	￥14,400
48回	￥11,300

XVIお買い上げの方に❶ニュージーランドストーリー ❷V-BALL ❸ジョイカード(連射式) ❹ディスケット20枚プレゼントいたします!!

現金でお買い上げの方には、さらに超特値でお出ししてます。ぜひ一度TEL下さい!!

上記組合せのディスプレイ(モニター)変更自由!!
詳しくは、お電話にてお問い合せ下さい!!

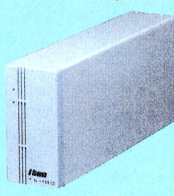
★XVI+HDD限定セット (送料・消費税込み)

① CZ-634C-TN
CZ-606D-TN
TX-130B
定価合計￥585,800
特価￥398,000

② CZ-634C
CZ-614D
TX-80B
定価合計￥688,000
特価￥468,000



+



周辺機器コーナー 電話で値切ろう。

プリンターセットコーナー

①CZ-8PC5 NEW 定価￥96,800
●48ドット●熱転写カラー漢字プリンター
大特価TEL下さい!!

②CZ-8PK10 (24ピン漢字プリンター136桁)
定価￥97,800…特価￥69,900

③CZ-8PG1 (24ピンカラー漢字プリンター80桁)
定価￥130,000…特価￥92,800

④CZ-8PG2 (24ピンカラー漢字プリンター136桁)
定価￥160,000…特価￥114,000

OAランド特選品!!



■IO-735XB (定価￥248,000)
●カラーイメージジェットプリンター ケーブル付 特価￥169,000

X68000用ハードディスク

■SCSIタイプ TOWNSでもOK
●アイテック
①TX-80S (￥108,000)… 特価TEL下さい
②TX-130S (￥138,000)… 特価￥ 94,500
③TX-180S (￥185,000)… 特価￥127,000

■SASIタイプ
●ロジテック
①SHD-40J (￥79,800)……特価￥ 55,000

※X68000SUPER/XVI以外の機種では、SCSIボードが必要となります。

★SCSIボード………特価￥ 22,000
★光ディスク………特価￥320,000
★JX220X…………特価￥120,000

X68000用周辺機器コーナー

①CZ-6VT1 (カラーイメージユニット)
定価￥69,800……特価￥ 51,500
②CZ-8NS1 (カラーイメージスキャナー)
定価￥188,000…特価￥135,000
③CZ-6BM1 (MIDIボード)
定価￥26,800……特価TEL下さい
④CZ-6BE2A (2MB増設RAMボード)
定価￥59,800……特価￥ 43,000
⑤CZ-6BE2B (2MB増設RAM)
定価￥54,800……特価￥ 39,800
⑥CZ-6BP2 (数値演算プロセッサ)
定価￥45,800……特価￥ 33,000
⑦CZ-6EB1 (拡張I/Oボックス=4スロット)
定価￥88,000……特価￥ 63,800
⑧CZ-6BP1 (数値演算プロセッサボード)
定価￥79,800……特価￥ 57,800

■ゲームソフト、ビジネスソフト2割・3割引きは、当り前で～す。お問い合せ下さい。

《計測技研》増設メモリ&プロセッサ

●高速増設メモリと数値演算プロセッサが一つのボードになった!!●

●KGB-X68PRKII-02 (￥ 55,000)……特価￥ 42,000
● PRKII-04 (￥ 90,000)……特価￥ 69,000
● PRKII-06 (￥125,000)……特価￥ 96,000
● PRKII-08 (￥160,000)……特価￥122,000
● PRKII-12 (￥ 85,000)……特価￥ 65,000
●KGB-X68PRKII-14 (￥120,000)……特価￥ 92,000
● PRKII-16 (￥155,000)……特価￥119,000
● PRKII-18 (￥190,000)……特価￥146,000
●MC-6888 IRC (￥38,000)…………特価￥ 28,500

I・Oデータ増設RAMボード



■PIO-6BE1-A (1MB) 定価￥25,000 特価￥16,000
■PIO-6BE2-2M (2MB) 定価￥50,000 特価￥31,500
■PIO-6BE4-4M (4MB) 定価￥88,000 特価￥55,000

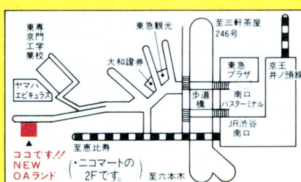
OAランド今月の特価品 在庫処分

CZ-634C-TN (2台限定)……特価￥260,000
CZ-644C-TN (2台限定)……特価￥365,000
CZ-623C (中古)……特価￥220,000
CZ-603C (中古)……特価￥150,000
CZ-611C+1MB……特価￥120,000

通信販売のご案内

全国通販

■銀行振込で申し込みの方は商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。
〔振込先〕第一勧業銀行 渋谷支店
普通No.1163457 ㈱オーエーランド



■年中無休です!!

■現金書留で送金されるお客様は電話番号と商品名、数量を明記して同封して下さい。■クレジットでご購入を希望される方は申し込み用紙をお送り致しますのでご記入の上返送して下さい。20才以上の方は、原則として保証人不要です。クレジットは1～60回払で月々5,000円よりご自由に設定できます。

クレジット表

3回	3.5%	6回	4.5%	10回	6%	12回	6%	15回	8.5%	18回	11%	20回	12%
24回	12.5%	30回	17%	36回	17.5%	42回	22.5%	48回	23%	54回	29%	60回	29.5%

㈱オー・エー・ランド

〒150 東京都渋谷区桜丘町3-13 アルカディア2F
☎(03)3770-8855

関東エリアの送料は、1個につき￥1,000です。 FAX(03)3770-7080

★全商品保証書付。専門のアドバイザーが、お客様のニーズに対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。

■本体・モニターのセットは、すべて送料・消費税込です。掲載の価格は、10月下旬現在です。

OFFICE KOUKAN

問答無用！新装開店！

全開電飾



出ます、出します、脱がせます！

ジャンジャンバリバリ
お時間の許す限り
お楽しみ下さい。





美人コンパニオンも多数お越しをお待ちしております。

X68000ならではのグラフィックによる美人コンパニオンが皆様の出玉によって変化に富んだ妖しい姿態でお待ちしています。

リアルなパチンコ台を数多くご用意致しました。

入賞率に変化のある各種モデルの違うパチンコ台を取り揃えました。玉の動きもリアルに本物のパチンコ気分で楽しめます。

音声、BGM、効果音も完備

FM音源8声プラスPCM音でサウンドも臨場感満溢。コンパニオンのおしゃべりも聞くことができます。

X68000版 好評発売中 ¥7,800（税抜）

PC-98版対応「満開電飾」も好評発売中!
(PC-9801VM21/UV2以降 各7,800円)

企画・制作：株式会社オフィス恒環　発売：ビクター音楽産業株式会社

通信販売　当社の商品をお近くのパソコンショップでお買い求めになれない場合、商品名、機種名、住所、氏名、電話番号を明記のうえ、下記住所まで定価プラス3%消費税分を現金書留にてお申し込み下さい（送料無料）〒151 東京都渋谷区千駄ヶ谷2-8-16 ビクター音楽産業㈱〈通信販売係〉

■店頭にて、新作ゲームソフト25～30%OFF!!（税別）、超低金利オクトハッピークレジットをご利用下さい!!

■冬のボーナス一括（12月・1月末）払いOK!!手数料無料!!ご利用下さい。■店頭にて、新作ゲームソフト25～30%OFF!!

パソコンプラザ OCT オクト

案内図

至品川　東口ロータリー　JR蒲田から徒歩5分　京浜蒲田から徒歩1分　至品川　三菱BK　JR蒲田駅　商店街アーケード　2F　京浜蒲田駅　富士BK　三和BK　メガネドラッグ　八幡神社　環八通り　〒　NTT　至横浜　至横浜

店頭セール実施中

オクトで始まるパソコンワールド

☎03-3730-6271

●営業時間 AM 11:00～9:00/日曜・祭日PM7:00　電話一本で、ハイ即納

〒144 東京都大田区蒲田4-6-7 FAX 03-3730-6273

全国通販

●定休日毎週火曜日 祭日の場合翌日になります。

オクト ラクラククレジット

3回	3.5	6回	4.5	10回	6.0	12回	6.0	15回	9.0	18回	11.0
20回	12.0	24回	12.5	30回	17.0	36回	17.5	48回	23.0	60回	33.0

OCT-1 システム インフォメーション

- ▶全商品保証付（メーカー保証）
- ▶超低金利ハッピークレジット（1回～60回）頭金ナシOK!
- ▶ボーナス一括払いOK!ボーナス2回払いOK!!
- ▶配達日の指定OK!（万全なサポート体制）
- ▶商品の組合せ自由!　オクトフリーダムシステム
- ▶店頭デモンストレーション実施中

オクト セレクテッドシステム



広告掲載商品以外の製品も取扱っております。

■便利です。夜9時まで営業しております。お立寄り下さい。お待ちしております!!

OCT-1 オクト 蒲田

冬のボーナス一括（12月・1月末）払いOK!!
ナナッ…ナント!手数料無料!!ウーーンお得!!

SHARP X68000 XVI エクシヴィ

- ■16MHz
- ■SX-WINDOW ver1.1
- ■Attachment MEMORY BORD

快速16MHz 鮮烈デビュー!!

■CZ-634C-TN（定価￥368,000）

Ⓐ●CZ-634C-TN ●CZ-614D-TN NEW
定価合計￥503,000 ▶特価TEL下さい。

12回	￥32,800	24回	￥17,400	36回	￥12,100	48回	￥9,500

Ⓑ●CZ-634C-TN ●CZ-607D-TN NEW
定価合計￥467,800 ▶特価TEL下さい。

12回	￥30,700	24回	￥16,300	36回	￥11,300	48回	￥8,900

Ⓒ●CZ-634C-TN ●CZ-606D-TN
定価合計￥447,800 ▶特価TEL下さい。

12回	￥29,200	24回	￥15,500	36回	￥10,800	48回	￥8,400

■CZ-644C-TN（定価￥518,000）

Ⓓ●CZ-644C-TN ●CZ-614D-TN NEW
定価合計￥653,000 ▶特価TEL下さい。

12回	￥42,600	24回	￥22,600	36回	￥15,700	48回	￥12,300

Ⓔ●CZ-644C-TN ●CZ-607D-TN NEW
定価合計￥617,800 ▶特価TEL下さい。

12回	￥40,400	24回	￥21,400	36回	￥14,900	48回	￥11,700

Ⓕ●CZ-644C-TN ●CZ-606D-TN
定価合計￥597,800 ▶特価TEL下さい。

12回	￥39,000	24回	￥20,700	36回	￥14,400	48回	￥11,300

X68000XVI ドッカ～ン!プレゼント!!

―あなたのオクトから素敵な贈物―

今、XVIをお買い上げいただいた方は、プレゼントの①番か②番のどちらかお選び下さい。プラス③番はもれなくプレゼント!!

① 大人気　新作!大人気ゲームソフト!!
生中継68 野球ゲームの決定版（定価￥9,800）
大戦略II（キャンペーン版）不朽の名作X68000版（定価￥9,800）

or

② インテリジェントコントローラ ■CZ-8NJ2（CYBER STICK）シューティングゲーマーの必須アイテム!!（定価￥23,800）

③ MD-2HD（10枚）シリコンキーボードカバー もれなく!!サービス!!

▶現金超特価 ￥TEL下さい!!▶

※どちらかお選び下さい!!（どっちが得かヨーク考えてネ!）

特選周辺機器（送料￥500）

- ●SX-68M MIDIインターフェースボード（システムサコム）￥19,800…特価￥13,600
- ●Fine Scanner X68（HAL研究所）（HGS-68）￥39,800…特価￥25,200

■増設RAMボード＝I・Oデータ

① PIO-6BE1-A（1MB）￥25,000…特価￥16,000
② PIO-6BE2-2M（2MB）￥50,000…特価￥31,800
③ PIO-6BE4-4M（4MB）￥88,000…特価￥55,000

周辺機器コーナー（送料無料）

商品	定価	特価
●CZ-6BEI IBM増設RAMボード	（￥35,000）	▶特価￥26,250
●CZ-6BEIB IBM増設RAMボード	（￥28,000）	▶特価￥21,000
●CZ-6BE2 2MB増設RAMボード	（￥79,800）	▶特価￥59,850
●CZ-6BE4 4MB増設RAMボード	（￥138,000）	▶特価￥103,500
●CZ-6BFI 増設用RS-232Cボード	（￥49,800）	▶特価￥37,350
●CZ-6BGI GP-IBボード	（￥59,800）	▶特価￥44,850
●CZ-6BMI MDIボード	（￥26,800）	▶特価￥20,100
●CZ-6BNI スキャナ用パラレルボード	（￥29,800）	▶特価￥22,350
●CZ-6BPI 数値演算プロセッサボード	（￥79,800）	▶特価￥59,850
●CZ-6BOI ユニバーサルI/Oボード	（￥39,800）	▶特価￥29,850
●CZ-6EBI/BK 拡張I/Oボックス	（￥88,000）	▶特価￥66,000
●CZ-6VTI/BK カラーイメージ・ユニット	（￥69,800）	▶特価￥52,350
●CZ-8NM2A マウス	（￥6,800）	▶特価￥5,100
●CZ-8NTI マウストラックボール	（￥9,800）	▶特価￥7,350
●CZ-8NSI カラーイメージスキャナ	（￥188,000）	▶特価￥140,000
●CZ-6BCI FAXボード	（￥79,800）	▶特価￥59,850
●CZ-8TM2 モデムユニット	（￥49,800）	▶特価￥37,350
●CZ-64H 増設ハードディスク	（￥120,000）	▶特価￥90,000
●CZ-6TU GY/BK RGBシステムチューナー	（￥33,100）	▶特価￥24,800
●BF-68PRO 高性能CRTフィルター	（￥19,800）	▶特価￥14,850
●CZ-6MOI 光磁気ディスクユニット	（￥450,000）	▶特価￥337,500
●CZ-6BSI SCSIインターフェースボード	（￥29,800）	▶特価￥22,350
●CZ-6BL2 LANボード	（￥298,800）	▶特価￥223,500
●CZ-6BVI（ビデオボード）	（￥21,000）	▶特価￥15,750
●CZ-6BE2A 2MB増設RAMボード	（￥59,800）	▶特価￥44,850
●CZ-6BE2B 2MB増設メモリ（チップ型）	（￥54,800）	▶特価￥41,100
●CZ-6BP2 数値演算プロセッサ	（￥45,800）	▶特価￥34,350
●AN-S100 スピーカーシステム（2本1組）	（￥36,600）	▶特価￥27,450

※クレジットの回数は1回～60回、ボーナス併用などありますのでお電話でお問合せ下さい。

■本体セット：送料無料（注）本体セット以外の周辺機器（プリンター、モデム、HDD等）及びソフトの送料は、北海道・九州地区＝1ケロ￥1500、■その他離島地区は、1ケロ￥2000となります。

※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは、電話でお問合せ下さい。

■特に人気のある商品によっては、しばらくお待ち願うことがありますのでご了承下さい!!

■超低金利クレジットをご利用下さい。1回～60回払い、頭金ナシ!!ボーナス1回及び2回払いOKです。

X68000 ラストチャンス!!

■SUPER／PROII／SUPER-HD

生中継68
野球ゲームの決定版
（定価￥9,800）

プレゼント

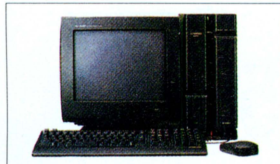
新作!大人気ゲームソフト!!

大戦略II（キャンペーン版）
不朽の名作X68000版
（定価￥9,800）

さらに! ★JOY CARD(連射式)×2個
さらにさらに!! ★MD-2HD 10枚

■ビッグバーゲンセール実施中!!ゲームソフト(ビジネス)新製品続々入荷中!!

限定



■SUPER（定価￥348,000）
CZ-604C-TN



■PROII（定価￥285,000）
CZ-653C-BK/GY



■SUPER-HD（定価￥498,000）
CZ-623C-TN

15型カラーディスプレイTV



CZ-614D-TN
定価￥135,000

14型カラーディスプレー



CZ-606D(GY/BK/TN)
定価￥79,800

21型カラーディスプレイ



CU-21HD
定価￥148,000

CZ-8NJ2 限定
●インテリジェントコントローラ
定価￥23,800
超特価￥18,000



Ⓐ CZ-604C＋CZ-614D……定価合計￥483,000▶￥306,000

12回	￥27,800	24回	￥14,700	36回	￥10,200	48回	￥8,000	60回	￥6,900

Ⓑ CZ-653C＋CZ-614D……定価合計￥420,000▶￥279,000

12回	￥25,300	24回	￥13,400	36回	￥9,300	48回	￥7,300	60回	￥6,300

Ⓒ CZ-623C＋CZ-614D 定価合計￥633,000▶￥366,000

12回	￥33,200	24回	￥17,600	36回	￥12,300	48回	￥9,600	60回	￥8,300

Ⓓ CZ-604C＋CZ-606D……定価合計￥427,800▶￥268,000

12回	￥24,300	24回	￥12,900	36回	￥9,000	48回	￥7,000	60回	￥6,100

Ⓔ CZ-653C＋CZ-606D……定価合計￥364,800▶￥218,000

12回	￥19,800	24回	￥10,500	36回	￥7,300	48回	￥5,700	60回	￥4,900

Ⓕ CZ-623C＋CZ-606D……定価合計￥577,800▶￥343,000

12回	￥31,200	24回	￥16,500	36回	￥11,500	48回	￥9,000	60回	￥7,800

Ⓖ CZ-604C＋CU-21HD 定価合計￥496,000▶￥313,000

12回	￥28,400	24回	￥15,100	36回	￥10,500	48回	￥8,200	60回	￥7,100

Ⓗ CZ-653C＋CU-21HD 定価合計￥433,000▶￥263,000

12回	￥23,900	24回	￥12,600	36回	￥8,800	48回	￥6,900	60回	￥6,000

Ⓘ CZ-623C＋CU-21HD 定価合計￥646,000▶￥373,000

12回	￥33,900	24回	￥18,000	36回	￥12,500	48回	￥9,800	60回	￥8,500

(送料無料・税別)★クレジット価格は、消費税込みですヨ～!ご利用下さい!!

X68000ソフト大セール実施中!!（ゲームソフト25～30%OFF） （送料￥500）

〈グラフィック〉●Z's STAFF PRO68K Ver.2.0
(シャフト)定価￥58,000
……………特価￥37,800

〈開発ツール〉●C-コンパラPRO68KV.2
定価￥44,800 CZ-245IS
……………特価￥32,800

〈データベース〉●CARD PRO68K Ver.2.0
定価￥29,800 CZ-253BS
……………特価￥20,800

〈グラフィック〉●C-TRACE 68 Ver.3.0
定価￥98,000
……………特価￥69,000

〈C言語〉●C & Professional Pack
定価￥58,000
……………特価￥40,000

〈音楽〉●Music studio PRO68K Ver. 2.0
定価￥28,800 CZ-261MS
……………特価￥21,200

〈CGシール〉●CANVAS PRO68K
定価￥29,800 CZ-249GS
……………特価￥22,200

〈ワープロ〉●Multiword PRO68K
定価￥32,000 CZ-225BS
……………特価￥23,800

〈通信〉●Tlepotion PRO68K
定価￥22,800 CZ-258BS
……………特価￥17,000

型名	商品	定価	特価
CZ-212BS	BUSINESS PRO-68K	￥68,000	￥48,000
CZ-213MS	MUSIC PRO68K	￥18,800	￥13,400
CZ-214MS	SOUND PRO-68K	￥15,800	￥11,400
CZ-215MS	Sampling PRO-68K	￥17,800	￥12,800
CZ-219SS	OS-9/X68000	￥29,800	￥21,000
CZ-220BS	DATA PRO-68K	￥58,000	￥41,000
CZ-223CS	Communication PRO-68K	￥19,800	￥14,200
CZ-224LS	THE 福袋 V2.0	￥9,900	￥7,500
CZ-241BS	システム手帳リフィル集	￥9,800	￥7,500
CZ-242BS	活用フォーム集	￥9,800	￥7,500
CZ-244SS	Homan 68K Ver2.0	￥9,800	￥7,500
CZ-247MS	MUSIC PRO-68K〔MIDI〕	￥28,800	￥20,800
CZ-240BS	Stationery PRO-68K	￥14,800	￥11,500
CZ-243BS	CYBER NOTE PRO-68K	￥19,800	￥15,200

型名	商品	定価	特価
	Z's TRIPHNY(デジタルクラフト)	￥39,800	￥27,300
	テラッツオ(ハミングバード)	￥19,400	￥13,800
	KAMIKAZE(サムシンググッド)	￥68,000	￥44,500
	Final Ver.3.2(エーエスピー)	￥38,000	￥29,500
	サイクロンEXPRESSα68	￥98,000	￥69,500
	Gツール(ザインソフト)	￥28,000	￥18,800
	たーみのる2(SPS)	￥17,800	￥13,200
	G68K Ver.2 PRO	￥22,000	￥17,500
CZ-259SS	SX-WINDOW Ver.1.0	￥6,800	￥5,000
CZ-251BS	ハイパーワード	￥39,800	￥29,600
CZ-260LS	XBAS to CHECKER PRO68K	￥9,800	￥7,500
CZ-234LS	AI-68K	￥188,000	￥139,000
CZ-255GS	CANVASドローグラフィックLiB	￥8,800	￥6,600
CZ-256GS	CANVASドローグラフィックVol.2	￥8,800	￥6,600

熱転写カラー漢字プリンター （送料￥1,000）

■CZ-8PC5



●48ドット
●熱転写カラー漢字プリンター
定価￥96,800
特価￥TEL下さい!!（ケーブル用紙付）

ハードディスク （送料￥1,000）

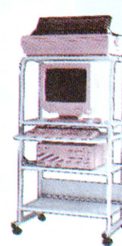
■アイテック

X68000用ハードディスク



● TX-80（定価￥108,000）……▶大特価￥77,000
(80MB、SCSI、SASI両対応)
● TX-130（定価￥138,000）…▶大特価￥95,000
(130MB、SCSI対応)
● TX-180（定価￥185,000）…▶大特価￥129,000
(180MB、SCSI対応)
※CZ-6BSI(SCSIボード)￥29,800‥▶特価￥22,350

パソコンラック〈送料無料〉



Ⓐ5段キャスター付
スライド式キーボード台
●1150(H)×640(W)×600(D)
定価￥38,000
特価 ￥12,800



Ⓑ4段キャスター付
●1250(H)×640(W)×700(D)
定価￥29,800
特価 ￥8,800

店頭新作ゲームソフト25～30%OFF!! ビジネスソフト25%より特価中

★通信販売お申込みのご案内★ 〒144 東京都大田区蒲田4-6-7 TEL:03-3730-6271

お申込みはお電話でお願いします。お客様の〈住所〉〈氏名〉〈電話番号〉及び〈商品名〉をお知らせ下さい。●入金確認後ただちに商品をご送付いたします。

現金一括払い
銀行振込：お近くの銀行より(電信扱い)にてお振込み下さい。
現金書留：封筒の中に住所・氏名・商品名をご記入の上当社までお送り下さい。

クレジット
専用お申込用紙をお送り致します。ので、必要事項をご記入、ご捺印の上ご返送下さい。手続きは簡単です。

オクト ラクラク クレジット表

3回	3.5	6回	4.5	10回	6.0	12回	6.0
15回	9.0	18回	11.0	20回	12.0	24回	12.5
30回	17.0	36回	17.5	48回	23.0	60回	33.0

振込先
富士銀行 久ヶ原支店 ㊟No.1824
三菱銀行 蒲田支店 ㊟No.0278691
株式会社 億人(オクト)

※掲載の価格は変動しますので、まずは、お電話にてご確認ください。
※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは電話でお問合せ下さい。
※銀行振込、または、現金書留でご注文の際には、あらかじめ電話でご確認の上、お申し込み下さい。

超低金利クレジットOK

注目!!

平成4年3月末一括払い
手数料(金利)無料

(平成3年12月末はもちろんのこと
平成4年1月末/2月末/3月末のいずれかをご指定下さい。)

またまた秋葉原で おなじみの

《増設メモリー&数値演算プロセッサ》計測技研

① PRKII-02(2M)	定価¥55,000	▶	特価¥41,900
② PRKII-04(4M)	定価¥90,000	▶	特価¥69,000
③ PRKII-06(6M)	定価¥125,000	▶	特価¥95,500
④ PRKII-08(8M)	定価¥160,000	▶	特価¥122,000
⑤ PRKII-12(2M)	定価¥85,000	▶	特価¥65,500
⑥ PRKII-14(4M)	定価¥120,000	▶	特価¥92,000
⑦ PRKII-16(6M)	定価¥155,000	▶	特価¥118,000
⑧ PRKII-18(8M)	定価¥190,000	▶	特価¥145,000
⑨ MC-68881RC	定価¥38,000	▶	特価¥28,000

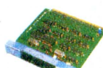
11/15~12/15

●お近くの方はお
●本体単品で特
●ビジネスソフト定

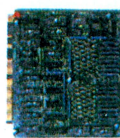
Fine Scanner-X68
(HAL研究所)X68000専用
■HGS-68(定価¥39,800)
特価¥25,300
(送料・消費税込み¥27,089)



X68000シリーズ専用
MIDIインターフェースボード
SX-68M(サコム)
(純生コンパチ)定価¥19,800
特価¥13,700
(送料・消費税込み¥14,626)



X68000メモリボード(シャープ & I/O・DATA)(送料¥500)



① CZ-6 BE1(600C用)定価¥35,000
(送料・消費税込み¥27,295)……特価¥26,000
② PIO-6BE1-A 定価¥25,000
(送料・消費税込み¥16,789)……特価¥15,800
③ PIO-6BE2-2M 定価¥50,000
(送料・消費税込み¥32,754)……特価¥31,300
④ PIO-6BE4-4M 定価¥88,000
(送料・消費税込み¥56,650)……特価¥54,500

限定 50台限り
■オムロン=モデム
● MD-24FP5II(MNP5)
定価¥42,800
▶P&A特価¥23,600
(送料・消費税込み¥25,338)

P&A超低金利クレジットをご利用ください!!

X68000-XVI

※クレジット表は、送料・消費税込み!!

XVI/XVI-HDセットでお買い上げの方に、もれなくプレゼント!!
①「熱血高校サッカー編(¥8,800)」
②「ダウンタウン熱血物語(¥8,800)」
はもちろん、さらにその上、人気の
イ「ロードス島戦記(¥9,800)」
ロ「パロディウス(¥9,800)」
ハ「生中継68(¥9,800)」
ニ「信長の野望武将風雲録(¥9,800)」
ホ「ELLE(エル)(¥7,800)」
の中のいずれか2本をプレゼント!!

注目!!



X68000-XVI ▶セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Ⓐセット:CZ-634C-TN+CZ-606D-TN…定価¥447,800▶特価価格はTEL下さい。

12回	29,100	24回	15,400	36回	10,600	48回	8,300	60回	7,000

Ⓑセット:CZ-634C-TN+CZ-614D-TN…定価¥503,000▶特価価格はTEL下さい。

12回	32,700	24回	17,200	36回	11,900	48回	9,300	60回	7,800

X68000-XVI-HD ▶セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Ⓐセット:CZ-644C-TN+CZ-606D-TN…定価¥597,800▶特価価格はTEL下さい。

12回	38,900	24回	20,500	36回	14,200	48回	11,100	60回	9,300

Ⓑセット:CZ-644C-TN+CZ-614D-TN…定価¥653,000▶特価価格はTEL下さい。

12回	42,300	24回	22,300	36回	15,500	48回	12,100	60回	10,200

※上記のモニターを、CZ-604D(定価¥94,800)、CZ-605D(定価¥115,000)、CU-21HD(定価¥148,000)に変更の場合、TEL下さい。超特価で販売致します。

X68000シリーズ~P&Aスペシャルセット

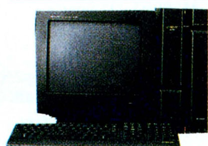
(送料¥2,000・消費税別)

注目!!

「P&Aスペシャルセット」にもれなくプレゼント!!
●上記XVI/XVI-HDのプレゼント
①、②+イ~ホの中の2本+さらにその上、目にやさしい。
Ⓒ「高性能CRTフィルター」(¥19,800)」又は、
Ⓓ「SX-WINDOW. Ver1.1」(¥9,800)
をプレゼント!!

※セットでお買い上げの方に、
●ディスケット10枚
●ジョイカード2個
プレゼント中!!

SUPER



Ⓐセット:P&A特選セット
■CZ-604C
(本体定価¥348,000)
+
■CZ-606D
(モニター定価¥79,800)
▶P&A超特価 ¥268,000

Ⓑセット
■CZ-604C+CZ-604D
定価¥442,800…▶特価¥275,000

Ⓒセット
■CZ-604C+CZ-607D
定価¥447,800…▶特価¥283,000

Ⓓセット
■CZ-604C+CZ-614D
定価¥483,000…▶特価¥306,000

Ⓔセット
■CZ-604C+CU-21HD
定価¥496,000…▶特価¥313,000

PRO-II



Ⓐセット:P&A特選セット
■CZ-653C
(本体定価¥285,000)
+
■CZ-606D
(モニター定価¥79,800)
▶P&A超特価 ¥218,000

Ⓑセット
■CZ-653C+CZ-604D
定価¥379,800…▶特価¥225,000

Ⓒセット
■CZ-653C+CZ-607D
定価¥384,800…▶特価¥233,000

Ⓓセット
■CZ-653C+CZ-614D
定価¥420,000…▶特価¥256,000

Ⓔセット
■CZ-653C+CU-21HD
定価¥433,000…▶特価¥263,000

SUPER-HD



Ⓐセット:P&A厳選セット
■CZ-623C
(本体価格¥498,000)
+
■CZ-606D
(モニター定価¥79,800)
▶P&A超特価 ¥328,000

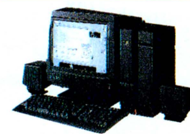
Ⓑセット
■CZ-623C+CZ-604D
定価¥592,800…▶特価¥335,000

Ⓒセット
■CZ-623C+CZ-607D
定価¥597,800…▶特価¥343,000

Ⓓセット
■CZ-623C+CZ-614D
定価¥633,000…▶特価¥366,000

Ⓔセット
■CZ-623C+CU-21HD
定価¥646,000…▶特価¥373,000

EXPERII



Ⓐセット:P&A厳選セット
■CZ-603C
(本体価格¥338,000)
+
■CZ-606D
(モニター定価¥79,800)
▶P&A超特価 ¥238,000

Ⓑセット
■CZ-603C+CZ-604D
定価¥432,800…▶特価¥243,000

Ⓒセット
■CZ-603C+CZ-607D
定価¥437,800…▶特価¥252,000

Ⓓセット
■CZ-603C+CZ-614D
定価¥473,000…▶特価¥277,000

Ⓔセット
■CZ-603C+CU-21HD
定価¥486,000…▶特価¥280,000

●本広告の掲載の商品の価格については、消費税は含まれておりません。●営業時間=平日AM10:00~PM7:00、日祭AM10:00~PM6:00

回～84回払いまでOK!!

★頭金なし!★即日発送

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前に、あらかじめお電話にてご確認下さい。

P&Aがズバリ超特価セールでご奉仕!!

立寄り下さい。専門係員が説明いたします。
価で受付します。詳しくは電話にてお問合せ下さい。
価の20%引きOK! TELください。

全国通販

超特価でクレジットが組める!!

X68000用ソフトコーナー（送料1ヶ～5ヶまで¥500・消費税別）

商品	定価	特価
●Z's STAFF PRO68K Ver.2.0（ツァイト）	定価¥ 58,000	特価¥ 37,500
●Z's TRIPHONY デジタルクラフト（ツァイト）	定価¥ 39,800	特価¥ 27,300
●テラッツォ（ハミングバード）	定価¥ 19,400	特価¥ 13,900
●KAMIKAZE（サムシング・グッド）	定価¥ 68,000	特価¥ 44,000
●C & Professional Pack（マイクロウェアジャパン）	定価¥ 58,000	特価¥ 40,000
●Final Ver3.2（エーエスピー）	定価¥ 38,000	特価¥ 29,200
●C-compiler PRO68K Ver.2 CZ-245LS	定価¥ 44,800	特価¥ 32,800
●CARD PRO68K Ver.2.0 CZ-253BS	定価¥ 29,800	特価¥ 22,900
●XBAS to C CHECKER CZ-260LS	定価¥ 9,800	特価¥ 7,400
●OS-9/X68000 CZ219SS	定価¥ 29,800	特価¥ 22,500
●HYPERWORD CZ-251BS	定価¥ 39,800	特価¥ 29,600
●THE 福袋 V2.0 CZ224LS	定価¥ 9,900	特価¥ 7,400
●SOUND PRO68K CZ-214MS	定価¥ 15,800	特価¥ 11,400
●MUSIC PRO68K CZ213MS	定価¥ 18,800	特価¥ 13,400
●Sampling PRO68K CD215MS	定価¥ 17,800	特価¥ 12,700
●MUSIC-studio PRO68K CZ-252MS	定価¥ 15,800	特価¥ 21,400
●MUSIC-PRO68K〔MIDI〕247MS	定価¥ 28,800	特価¥ 20,700
●New-print Shop 221HS	定価¥ 19,800	特価¥ 15,500
●Communication 223CS	定価¥ 19,800	特価¥ 14,200
●Communication Ver.2 CZ-257CS	定価¥ 19,800	特価¥ 15,500
●C-TRACE68 Ver.3.0（キャスト）	定価¥ 98,000	特価¥ 69,000
●サイクロンEXPRESSα68	定価¥ 98,000	特価¥ 69,800
●G68K Ver2 PRO	定価¥ 22,000	特価¥ 17,500
●SX-WINDOW CZ-259SS	定価¥ 6,800	特価¥ 4,900
●Gツール（ザインソフト）	定価¥ 28,000	特価¥ 18,900
●たーみのる2（SPS）	定価¥ 17,800	特価¥ 13,300
●マジックパレット（ミュージカルプラン）	定価¥ 19,800	特価¥ 14,500
●Hyper word CZ-251BS	定価¥ 39,800	特価¥ 29,600

●ゲームソフト20%OFF OK!!（一部ソフト除く）

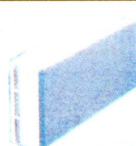
周辺機器コーナー（送料¥500・消費税別）

	商品	定価	特価
1	CZ-8NSI	定価¥188,000	特価¥136,000
2	CZ-6VTI	定価¥ 69,800	特価¥ 51,500
3	CZ-6TU	定価¥ 33,100	特価¥ 24,300
4	BF-68PRO	定価¥ 19,800	特価¥ 14,900
5	CZ-6BEI	定価¥ 35,000	特価¥ 26,000
6	CZ-6BEIA	定価¥ 38,000	特価¥ 28,500
⑦	CZ-6BE2A	定価¥ 59,800	特価¥ 43,500
⑧	CZ-6BE2B	定価¥ 54,800	特価¥ 39,800
9	CZ-6BFI	定価¥ 49,800	特価¥ 37,800
10	CZ-6BPI	定価¥ 79,800	特価¥ 59,800
11	CZ-6BMI	定価¥ 26,800	特価¥ 19,800
12	CZ-6EBI	定価¥ 88,000	特価¥ 65,600
13	AN-S100	定価¥ 36,600	特価¥ 26,800
14	CZ-6SDI	定価¥ 44,800	特価¥ 35,000
15	CZ-6BN1	定価¥ 29,800	特価¥ 22,600
16	CZ-6BV1	定価¥ 21,000	特価¥ 15,900
17	CZ-64H	定価¥120,000	特価¥ 91,500
18	CZ-6BG1	定価¥ 59,800	特価¥ 45,000
19	CZ-6BU1	定価¥ 39,800	特価¥ 30,300
20	CZ-6PVI	定価¥198,000	特価¥153,000
21	CZ-6BS1	定価¥ 29,800	特価¥ 22,300
22	CZ-8NJ2	定価¥ 23,800	特価¥ 18,500
23	CZ-6BL2	定価¥298,000	特価¥222,000
24	JX-100S	定価¥ 89,800	特価¥ 48,500
㉕	JX-220X	定価¥168,000	特価¥126,000
㉖	IO-735XB	定価¥248,000	特価¥169,000

（IO-735XBご購入の方「BANANA-PRINT」プレゼント!!

X68000用ハードディスク（送料¥1,000）

アイテック



■TX-80（80MB）……定価¥108,000▶特価¥ 77,000
（SCSI・SASI両用）（送料・消費税込み¥80,340）

■TX-130（130MB）……定価¥138,000▶特価¥ 97,000
（SCSI）（送料・消費税込み¥100,940）

■TX-180（180MB）……定価¥185,000▶特価¥131,000
（SCSI）（送料・消費税込み¥135,960）

プリンター（ケーブル・用紙付）（送料¥1,000・消費税別）



■CZ-8PC5-BK NEW……定価¥ 96,800▶特価価格はTEL.!!
■CZ-8PK10……定価¥ 97,800▶特価¥71,000
■CZ-8PG2……定価¥160,000▶特価価格はTEL.!!
■CZ-8PG1……定価¥130,000▶特価価格はTEL.!!

モデムコーナー（送料¥1,000）

■COMSTARZ CLUB24/5
（NEC）定価¥39,800
特価¥26,300（送料・消費税込み ¥28,119）

■MD-24FB5V
（オムロン）定価¥39,800
特価¥26,300（送料・消費税込み ¥28,119）

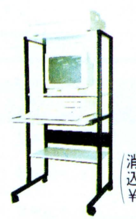
P&A特選パソコンラック（消費税別）（送料無料）

①3段¥7,900



消費税込み ¥8,137

②4段¥8,800



消費税込み ¥9,064

③5段¥12,800



消費税込み ¥13,184

全機種＝移動自由（キャスター付）・キーボード収納可（5段のみ）＝1230（H）×600（D）×650（W）

中古パソコンはP&Aにおまかせ!!

その場で高価現金買取り・高価下取りOK!!

■まずはお電話下さい。
03-3651-1884、FAX:03-3651-0141

■下取り・買取りでお急ぎの方、直接当社に来店、または、宅急便にてお送り下さい。

●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。（差額は、P&A超低金利クレジットをご利用下さい。）

●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。

●近郊の方は、P&A本店まで、直接お持ち下さい。即金にて、¥1,000,000までお支払い致します。

《便利な超低金利クレジットをご利用下さい》

●月々¥1,000円からOK!! ●ボーナス払いOK（夏冬10回までOK）
●支払い回数 1回～84回 ●お支払いは、8ヶ月先からでもOK!!

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
（電信扱いでお振込み下さい。）

〔振込先〕 住友銀行 新小岩支店
普通預金 1451576 ㈱ピー・アンド・エー

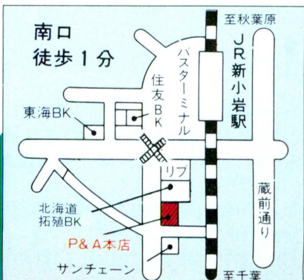
〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回～84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

アフターサービス万全
全商品保証付。専門の担当者がお客様の立場で対応します。
初期不良、輸送トラブルetc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

●定休日/毎週水曜日＝第3水曜（祭日の場合は翌日になります）

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	3.0	4.0	5.5	5.5	8.5	11.5	16.0	21.0	27.0	33.0



マイコン専門ショップ P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-3651-0148（代） FAX.03-3651-0141
営業時間
平日：AM10:00～PM7:00
日祭：AM10:00～PM6:00

●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

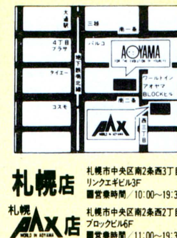
【各店のお休み】
12月のお休み／5日(木)・12日(木)・19日(木)
11月のお休み／7日(木)・14日(木)・21日(木)・28日(木)





池袋本店
豊島区東池袋1-28-1
■営業時間/11:00~19:00





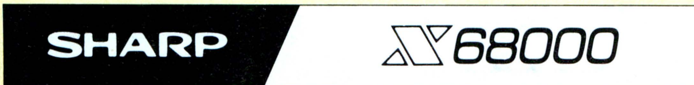
札幌店 札幌市中央区南2条西3丁目 リンクエギビル3F 営業時間/10:00~19:30
札幌 AAX店 札幌市中央区南2条西2丁目 ブロックビル6F 営業時間/11:00~19:30



在庫もBIGスケール・価格BIGにプライスダウン!!

■**札幌店**／札幌市中央区南2条西3丁目 リンクエギビル3F ☎011-251-1777 〈営業時間〉11:00~19:30
■**札幌 AAX店**／札幌市中央区南2条西2丁目 ブロックビル6F ☎011-251-5315 〈営業時間〉11:00~19:30
■**池袋本店**／東京都豊島区東池袋1-28-1 ☎03-3989-1171 〈営業時間〉11:00~18:30

電話でのご注文の場合
(03)3987-7771
お電話番号はおかけ間違いのないようにお願いします。
●北海道受注センター 011-251-6771
●九州受注センター 092-716-7771



X68000万全のサポート

AOYAMAにて購入のX68000は万一故障の場合でも全国どこでも出張サービスがうかがいます。 万一の場合ワールドインアオヤマサポート係にお電話下さい。お客様のお名前と電話番号だけで手続きは完了。



CZ-634C-TN
CZ-634C-TN〔2M本体16MHz〕……¥368,000
CZ-607D-TN〔.31 14インチチューナー付〕¥99,800
3Mフロッピーディスケット……¥9,000
定価合計¥476,800➡現金特価
クレジットは、お電話にて御問い合わせください

X68000 CZ-634C-TN
CZ-634C-TN〔2M本体16MHz〕……¥368,000
CZ-614D-TN〔.31 15インチチューナー付〕¥135,000
3Mフロッピーディスケット……¥9,000
定価合計¥512,000➡現金特価
クレジットは、お電話にて御問い合わせください

CZ-634C-TN
CZ-634C-TN……¥368,000
CZ-606D-TN……¥79,800
AP-900……¥92,800
プリンターケーブル……¥4,800
定価合計¥550,400➡¥369,800
クレジットは、お電話にて御問い合わせください

CZ-644C-TN
CZ-644C-TN〔2M本体16MHz80MHDD〕……¥518,000
CZ-614D-TN〔.31 15インチチューナー付〕……¥135,000
3Mフロッピーディスケット……¥9,000
定価合計¥662,000➡現金特価
クレジットは、お電話にて御問い合わせください

SX68M
SX68M……¥19,800
CM32L……¥69,000
MA12AV×2……¥28,000
定価合計¥117,600➡¥94,000
クレジットは、お電話にて御問い合わせください

CZ-653C-BK / CZ-606D-BK
¥218,000
CZ-653C-BK〔1M本体〕……¥285,000
CZ-606D-BK〔.31 14インチカラーCRT〕……¥79,800
定価合計¥382,400➡¥226,000

CZ-634C-TN / CZ-603D
¥299,000
CZ-634C-TN……¥368,000
CZ-603D〔限定12台〕……¥79,800
定価合計¥447,800➡¥299,000

CZ-604C-TN / CZ-606D-TN
¥268,000
CZ-604C-TN……¥348,000
CZ-606D-TN……¥79,800
住友3M 5"2HD……サービス品
定価合計¥427,800➡¥268,000

SHARP X68000 PRO II・PRO II HD PERSONAL WORKSTATION
CZ-653C-BK 中古品
¥162,000
定価合計¥285,000➡¥162,000

SHARP X68000 XVI PERSONAL WORKSTATION
CZ-634C-TN 中古品
¥258,000
定価合計¥368,000➡¥258,000



X68000ソフト＆周辺機器

X68000をはじめソフト&周辺機器類は、当社池袋店・札幌店・旭川店・福岡店にて実演中です。各店X68000コーナーが常設されております。

システムサコムSX-68M	MIDIボード	¥19,800➡¥15,250
アイテック TX-80	80MB HDD	¥108,000➡¥80,000
I/Oデータ PIO-6BE1A	IMB増設RAM	¥25,000➡¥17,800
SHARP マルチワード	マルチワープロソフト	¥32,000➡¥24,000
SHARP Ccompiler PRO-68K	Cコンパイラ	¥44,800➡¥33,600
システムサコム Mu-1 Suqer	MIDI用ソフト	¥39,800➡¥29,800
SHARP CZ-8PC5	80桁熱転写プリンタ	¥94,800➡¥69,800
SHARP IO-735X	136桁インクジェットプリンタ	¥248,000➡¥168,000
アイテック TX-130	130MB HDD	¥138,000➡¥111,000
ハル研 HGS-68	ファインスキャナー68	¥39,800➡¥29,800
SHARP CZ-6BE1	CZ-600C専用IMB増設RAM	¥35,000➡¥26,800
SHARP CZ-6BE1B	IMB増設RAM	¥28,000➡¥21,800
SHARP JX-220XB	イメージスキャナ	¥168,000➡¥134,400
SHARP CZ-8NJ2	インテリジェントコントローラー	¥23,800➡¥18,800
ローランド MT-32	MIDI音源	¥64,000➡¥49,800
SHARP CZ-8PK10	136桁ドットプリンター	¥97,800➡¥70,000
SHARP BF-68PRO	テレビフィルター	¥19,800➡¥14,800
SHARP CZ-6BM1	MIDIボード	¥26,800➡¥19,800
アイテム X Stor40	HDD	¥118,000➡¥89,800

全国出張サポート★ 私共にてご購入いただいたX68000は全国出張サポートがうけられます。

超お買得品（本体・ディスプレィ・プリンター・周辺） 詳しくは 03-3987-7701

CZ-634C-TN……¥368,000➡¥262,000
CZ-644C-TN……¥518,000➡¥368,000

X68000シリーズ
CZ-600C〔本体〕……¥369,000➡¥150,000
CZ-601C〔本体〕……¥319,000➡¥168,000
CZ-611C〔本体:20MB, HDD付〕……¥399,000➡¥188,000

CZ-652C〔本体:1MB68000PROタイプ〕……¥298,000➡¥160,000
CZ-604C〔本体2MB〕……¥348,000➡¥198,000
CZ-603C……¥338,000➡¥220,000
CZ-613C……¥448,000➡¥278,000
CZ-653C……¥285,000➡¥162,000
CZ-602D〔0.39カラーディスプレーテレビ〕……¥99,800➡¥64,000
CZ-604D〔0.31カラーディスプレー〕……¥94,800➡¥59,800

CZ-605D〔0.39カラーディスプレーテレビ〕……¥115,000➡¥78,000
CZ-613D〔0.31カラーディスプレーテレビ〕……¥135,000➡¥80,000
CZ-606D〔0.31ディスプレー〕……¥79,800➡¥55,800

周辺機器
SHD40〔X68000 40M HDD〕〔新品限定品〕……¥99,800➡¥63,000
TX-80〔X68000 80M HDD〕〔新品限定品〕……¥108,000➡¥88,000

サポート

万一のときも完全バックアップ。

万一の初期不良があった場合でも当社では万全の体製でお客様をフォロー致します。通常の初期不良ワクを大きく拡げ、最長1ヶ月間まで新品との交換を致しております。
また一週間以内の不良の場合は、こちらからお荷物をひきとりに伺います。
※ソフト及び中古商品に関しましては1ヵ月サービスの対象外とさせて頂きます。

グーンとお得な下取りシステム。
ゆうゆうお支払いは8ヶ月先から。
学生の味方、キャンバスクレジットがますますワイドに。
お支払いはナント！84回まで。
業界一番のスーパークレジットで。

03-3987-7795
すでにご注文いただいているお届け時間(時期)やメンテナンス、その他のお問い合せは上記へお電話下さい。

ファクシミリでご利用の場合
03-3985-5221
●ご注文方法(黒色のボールペン、またはサインペンでご記入下さい。)
①電話番号・住所・氏名又はお客様番号、お支払い方法をご記入下さい。

●銀行振込みの場合

取引銀行	住友銀行 池袋支店
口座番号	普通 1065392
口座名	株式会社 ワールドイン アオヤマ

AOYAMA WORLD IN
FOR THE EVOLUTION OF YOURLIFE

信頼と安さの TSUKUMO

'91年末日までGO!!

1991 FINAL DASH SALE

掲載商品2万円以上送料無料(離島を除く)

毎年恒例！秋葉原電気まつり

賞金総額7,000万円 11/22(金)〜1/7(火)

ますます好評の秋葉原電気まつり。秋葉原の店頭でお買物をされたお客様、商品代金¥5,000毎に抽選券を1枚さし上げます。

- 1等＝現金10万円
- 2等＝現金5万円
- 3等＝現金3万円
- 4等＝現金5千円

FINAL DASH ディスケットプレゼントセール 12月1日〜12月31日迄

期間中にツクモTSシリーズのTSドライブ(商品代金2万円以上)お買い上げのお客様にディスケット3.5インチ=5枚or5インチ=10枚をプレゼント.!!

新製品の情報もいろいろな機種の情報も、ツクモに行けばあなたのもの。歩きまわる必要はありません。ツクモだけで十分です.!!

★★★★★★★★★★シャープ製品は、小さいモノはポケコンから大きいモノは液晶ビジョンまで、何でも揃う！★★★★★★★★★★

イベントのお知らせ 「シャープわんさかバザール」開催

毎年恒例のシャープフェアを開催致します。東京各店にて催し物や特価品等を数多く取り揃えお待ちしております。ぜひお出かけ下さい。

開催日：11月23日(土)・24日(日)

場所：●ツクモパソコン本店●ツクモAVカメラ館 ●ツクモニューセンター店

新製品32bit NOTE型AXパソコンお取り扱い中!!

ツクモ X68000用TSドライブ

目のつけどころがツクモでしょ。

X68000シリーズ用3.5インチフロッピーディスクドライブ 大好評につき在庫お問合せ下さい。

- 3.5インチ2DD/2HD対応ドライブ使用。
- 2DD用ディバイスドライバ付属。
- 1.44MBディバイスドライバ付属。

3.5インチ1ドライブ TS-3XR1 定価¥44,800 ツクモ特価¥35,800 (消費税別途¥1,074)

3.5インチ2ドライブ TS-3XR2 定価¥57,800 ツクモ特価¥46,800 (消費税別途¥1,404)

やっぱりXVI これが一番!

X68000 XVI 快速16MHz

- CPUクロック周波数スピードアップ(16MHz)
- 増設メモリ本体内蔵可能(8MBまで)
- NEW SX-WINDOW搭載

■X68000XVI(CZ-634C-TN) 標準タイプ……定価¥368,000

■X68000XVI-HD(CZ-644C-TN) HD内蔵タイプ……定価¥518,000

お買得ハードディスクセット

- CZ-634C-TN……¥368,000
- CZ-614D-TN……¥135,000
- TX-130……¥138,000

合計定価¥641,000

ツクモ特価¥488,000 (消費税別途¥14,640)

クレジット例(48回払・税込) 初回¥16,418+月々¥13,500×47回

(買い換え・下取りも取り扱っております。是非、お尋ね下さい。)

ツクモニューセンター店☎03-3251-0987

X68000シリーズ用オプションボード

品名	特価	品名	特価
1MB増設RAMボード(CZ-600C専用)	特価¥20,000	GP-IBボード	特価¥50,800
1MB増設RAMボード(ACE/PRO/PRO2シリーズ用)	特価¥17,500	増設RS-232Cボード	特価¥42,300
2MB増設RAMボード(拡張スロット専用)	特価¥34,800	スキャナ用パラレルボード	特価¥25,300
4MB増設RAMボード(拡張スロット専用)	特価¥61,500	ユニバーサルI/Oボード	特価¥33,800

※計測技研のメモリーボードも取り扱っております。価格についてはお尋ね下さい。

★★★★★★ X68000用ハードディスク ★★★★★★

80MB SCSI/SASI両対応タイプ **TX-80** 定価¥108,000 特価¥76,000 SCSIボードセット¥100,000

130MB SCSI対応タイプ **TX-130** 定価¥138,000 特価¥96,000 SCSIボードセット¥120,000

180MB SCSI対応タイプ **TX-180** 定価¥185,000 特価¥130,000 SCSIボードセット¥154,000

大容量記憶装置

SCSIタイプHDDの場合、本体がSUPER/XVI以外の場合にはSCSIボード(CZ-6BS1)が必要です。

更に大容量が欲しい方は… ツクモはSONY MOの認定店です。

SONY光磁気ディスクユニットセット

NWP-539N(光磁気ディスクドライブ) SCSIケーブル 光磁気カートリッジ SCSIインターフェースボード

ツクモ特価¥398,000

シャープ純正「CZ-6MO1」も特価販売中!

電子手帳

- ■ハイパー電子システム手帳 PA-9500……定価¥48,000 ツクモ特価¥43,000
- PA-9550……定価~~¥59,000~~
- ■スタイリッシュ電子システム手帳 PA-X1 定価¥29,800 ツクモ特価¥26,000
- ■PA-S1 NEW……定価~~¥220,000~~
- ■Teleportion PRO68K……定価~~¥22,800~~
- ■CE-300L電子手帳通信ケーブル……ツクモ特価¥2,380

アートツール(ハード)

- ■JX-220X A4サイズカラーイメージスキャナー……定価~~¥168,000~~
- ■HGS-68 ファインスキャナーX68 ツクモ特価¥29,800
- ■CZ-6VT1 カラーイメージユニット……定価~~¥69,800~~
- ■CZ-6BV1 ビデオボード……定価~~¥21,000~~
- ■XAV-1Sアナログ RGB・S端子変換ユニット ツクモ特価¥6,890
- ■CZ-8PC5 48ドットカラー漢字熱転写プリンター……定価~~¥96,800~~

開発ツール

- ■C Compiler PRO-68K Ver2.0 定価~~¥44,800~~
- ■XBAS TO C CHECKER PRO-68K 定価~~¥9,800~~

ビジネスツール

- ■Multiword NEW……定価~~¥32,000~~
- ■FIXER Ver4.0……ツクモ特価¥15,800
- ■CARD PRO-68K Ver2.0 NEW 定価~~¥29,800~~

パソコン通信

- ■モデム 2400ボー/MNP5&V42bis対応 ツクモ特価¥29,800
- ■通信ソフト た〜みのる2 ツクモ特価¥14,240
- ■通信ソフト Teleportion PRO-68K 定価~~¥22,800~~

アートツール(ソフト)

- ■CANVAS PRO-68K……定価~~¥29,800~~
- ■Easy Paint SX-68K(CZ-263GW) 定価~~¥12,800~~
- ■NEW PrintShop PRO-68K Ver2.0……定価~~¥20,000~~
- ■Z's STAFF PRO-68K Ver2……ツクモ特価¥46,400
- ■マジックパレット……ツクモ特価¥15,800

コンピュータミュージック(X68000用)

NEW Aセット

- CM-32L……¥69,800
- SX-68M-II……¥19,800
- Musicstudio Mu-1 Ver1.4……¥19,800

合計定価¥108,600

ツクモ特価¥88,000 (消費税別途¥2,640)

クレジット例(18回払・税込) 初回¥7,223+月々¥5,600×17回

NEW Bセット

- CM-300……¥58,000
- SX-68M-II……¥19,800
- Mu-1 SUPER……¥39,800

合計定価¥117,600

ツクモ特価¥92,000 (消費税別途¥2,760)

クレジット例(10回払・税込) 初回¥10,967+月々¥10,100×9回

NEW Cセット

- CM-500……¥115,000
- SX-68M-II……¥19,800
- Mu-1 SUPER……¥39,800

合計定価¥174,600

ツクモ特価¥141,000 (消費税別途¥4,230)

クレジット例(15回払・税込) 初回¥17,079+月々¥10,600×14回

NEW Dセット

- CM-64……¥129,000
- SX-68M-II……¥19,800
- Mu-1 SUPER……¥39,800

合計定価¥188,600

ツクモ特価¥154,000 (消費税別途¥4,620)

クレジット例(18回払・税込) 初回¥10,940+月々¥9,900×17回

※この他の組み合わせは、お問い合わせ下さい。☎03-3251-9911へ

ローランド 追加オプション機器

- ステレオマイクロモニター CS-10……定価~~¥17,000~~
- MIDIキーボードコントローラー PC-200……定価~~¥36,000~~
- はなうたくん CP-40……定価~~¥33,000~~

※本格的MIDIは7号店2F MIDIフロア☎03-3253-4199へ

超低金利 冬・夏ボーナス2回払受付中!! 詳しくは☎03(3251)9911、又は各店へ

商品のご注文は通販受注専用センター フリーダイヤル **0120-377-999**

商品についての詳しいお問い合わせは各店、又は☎03(3251)9911へ

秋葉原各店 営AM10:15〜PM7:00

パソコン本店 5号店 7号店 ニューセンター店 AV/カメラ館

ツクモは「スーパーX PRO SHOP」です。

PRO STAFF ツクモ

九十九電機株 〒101-91 東京都千代田区神田郵便局私書箱135号

★商品のご注文は在庫確認の上お願いします。★表示価格には消費税は含まれておりません。

- ■ツクモパソコン本店2F ☎03-3253-5599(担当/荒井) 休毎週木曜
- ■ツクモニューセンター店 ☎03-3251-0987(担当/福地) 休毎週木曜 (12月は無休・正月は1/2から但しニューセンター店は1/4から)

便利で安心な通信販売 ツクモ通販センター☎03-3251-9911

- ■ツクモ5号店 ☎03-3251-0531(担当/森) 休毎週木曜 (12月は無休・正月は1/4から)
- ■ツクモAV/カメラ館B1 ☎03-3254-3999(担当/川名) 休毎週水曜 (12月は無休・正月は1/2から)
- ■名古屋1号店 ☎052-263-1655(担当/吉高) 休毎週月曜 (12月より1/8迄無休)
- ■名古屋2号店 ☎052-251-3399(担当/横山) 休毎週水曜 (12/12より1/8迄無休)
- ■ツクモ札幌店 ☎011-241-2299(担当/田口) 休毎週水曜 (12月は無休・正月は1/2から)

安心 迅速 高額 買い取りの ツクモニューセンター店

買い取り一発買取り窓口 電話受付(AM11:00〜PM5:00) ☎03(3251)9977 FAX受付(24時間) ☎03(3251)0299

ツクモグローバルカード

大人気/入会者募集中/ 18才以上なら学生でもOK。

お申し込みは☎03(3251)9898又は店頭にて/

※各店頭では、JCB・日本信販・DC・セントラル・マスター他各種カードも取り扱っております。

カード払い	全国代金引き換え配達	クレジット払い	現金書留払い	銀行振込払い	各種リース払い
通信販売での御利用カード、ツクモグローバルカード、VIPカードセントラル、ジャックス・御本人様より電話で通信販売部へお申し込み下さい。	お申し込みは☎03-3251-9911へお電話1本 配達日の指定もできます。	月々¥3,000以上の均等払いも頭金なし。夏・冬ボーナス2回払いも受付中!!	〒101-91 東京都千代田区神田郵便局私書箱135号 ツクモ通販センター oh!X係	事前に☎でお届け先をご連絡下さい。三和銀行 秋葉原支店 (普)1009939 ツクモデンキ	くわしくは各店にお問い合わせ下さい。ケースに合わせてご相談にのります。

◆◆◆◆企業の方へ…お見積りはFAXで。ツクモパソコン本店FAX03-3253-5199担当/荒井へ◆◆◆◆

THE DOWS

SOFT BANK

特別付録
5"2HDディスク

WINGZ
体験飛行



'91新創刊第3弾
来たる12月7日

DOS/Vマガジン

創刊

ハードの壁を越えた高品位パソコン
DOS/Vマガジン

12月8日創刊

毎月8日発売

A4変型判／本文160頁

予価780円



発行：ソフトバンク出版事業部

ELECTRONICS SHOW '91 & DATA SHOW '91

エレクトロニクスショウ

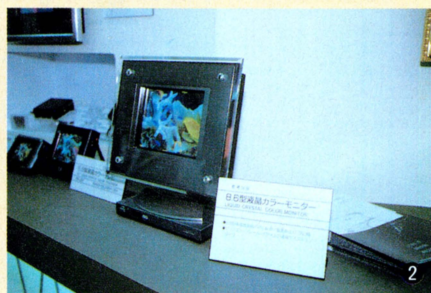
実用へ急ピッチのハイビジョン，ついに発売なるかCD-I，次世代オーディオはDCC(Digital Compact Cassette）かミニディスクか？ CS放送のPCM音声は？ ……と，AVマニアには話題の多いエレショウ。今回は10月1日から5日間，幕張メッセで開催された。

最近はアスペクト比を変えた横長テレビが話題になっているが，本物のハイビジョンと並べると差は歴然。ハイビジョンは0距離でも画像が破綻しない。大画面ほど違いは大きい。

圧巻は日立のハイビジョンプロジェクタ。解像度であるとか，明るさとか，色ズレ，視認画角の狭さといったプロジェクタの欠点がことごとく改善されている。プロジェクタっぽくない画質のプロジェクタだ。

そしてCD-I。家電数社からプレイヤーが出展され，ソフトも制作が進んでいる。発売はもう目前に迫っている。

















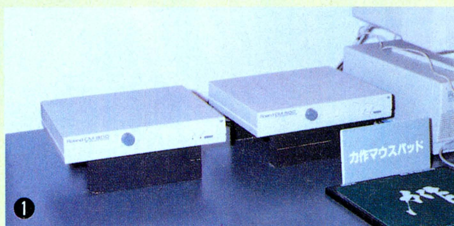
データショウ

10月22日から25日までの4日間にわたって，データショウ'91が晴海国際展示場で開催された。ネットワーク，カラー化が進むノートパソコンなどコンピュータ関連の出展も数多い。

パソコンを見ると非主流国産メーカーの多くがAXとOADG（Open Architecture Development Group）仕様を並列するようになってきている。OADGへ流れは変わりつつある。現状はともかく，今後主流になると思われるWINDOWSやNETWAREでは圧倒的にIBM互換機が優位にある。このまま大きな流れになるか，国民機の巻き返しは可能かどうかが注目される。

DTPではMacintoshの独壇場といっていい。しかし，アプリケーションが重すぎるため永らく高速機が待望されていた。今回68040を使ったQUADRAシリーズが加わって，独占状態にいっそう拍車がかかることが予想される。逆にいえば，16ビットのMacintoshはもう使いものにならないということでもあるのだが……。

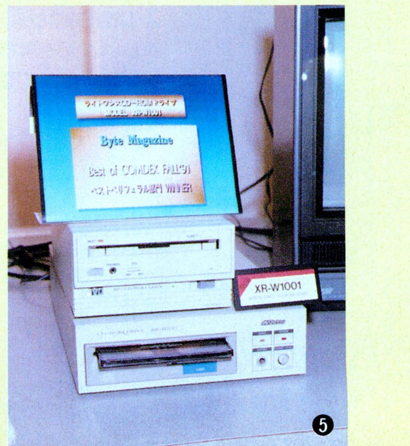
そして大容量外部記憶装置。本命は198,000円の3.5インチ光磁気ディスクドライブ，ICMのMO-3120だ。3000rpmの高速タイプで120Mバイトの容量を持つ。PC-9801用だがインタフェイスはもちろんSCSI。X68000につながるかな？





















❶シャープブース。小さくX68000が見える
❷8.6型の液晶テレビ
❸液晶ミュージアムの額縁風バリエーション
❹TFTカラー液晶を使ったノートパソコン
❺日立の大型ハイビジョンプロジェクタ。ストロボ撮影でも鮮明だ
❻横長のハイビジョンを3つ使ったシステム
❼ハイビジョンビデオプリンタ。画質は写真と同等
❽横長テレビ。コンバータでハイビジョン放送を表示できる
❾同じ発想を普通のテレビサイズでやる三菱のモニタ。非ハイビジョンでは最高の画質
❿子供用のCD-Iソフト。ポータブルプレイヤーを使用している
⓫これは据え置き型CD-I。ハワイの観光案内をしているところ
⓬CDRは書き込み可能なCD。上に乗っているのは普通のCDプレイヤー。当然デジタルで接続
⓭ポータブルDCCプレイヤー。これまでのカセットテープも再生できる
⓮ミニディスクは音楽用光磁気ディスクだ
⓯ホンダのビート……ではなくて車載テレビ
⓰こっちにもビートとおねえさん。出展物はナビゲーションシステム
⓱ジャイロを使った平行センサのデモ。ラジコンで自転車が走る！ 自転車は商品化されないだろうか








❶GS対応音源，RolandのCM-300/500
❷シャープのカラーノート。386SLを採用
❸DATを使った画像ストレージシステム
❹ICMの3.5インチ光磁気ディスクドライブは定価198,000円！
❺1回だけ書き込めるCD-ROMドライブ
❻新しいポータブルMacintosh
❼68040採用のQUADRA700/900
❽手書きそのままの毛筆プロッタ。結構器用に動く
❾フォントウェーブは書体倶楽部に代わるアウトラインフォントの標準になるか？
❿小さなトラックボール。同様なデバイスでボールを押すとボタンクリックになるタイプもある
⓫謎のファジィLUNA

[第7回]
プレゼント

寺尾 響子

響子inCGわ〜るど

扉のまえに立っていました。開けるには，鍵を使うか呪文を唱えるかしなくてはなりません。何もないし，何も知らない。考え込んでいると，向こう側から，

「どうぞ」

という声がしたので，ノブを回すと簡単に開きました。霧が立ち込めたように全体が真っ白で，その中に黒い点がぽつぽつ無数に散っています。一歩踏み込む。すると，外に出てしまいました。くるりと向きを変えて，もう一度トライ。また外へ出てしまいました。何回やっても同じことの繰り返し。ルーチン・ワーク。あきらめて扉を閉めます。

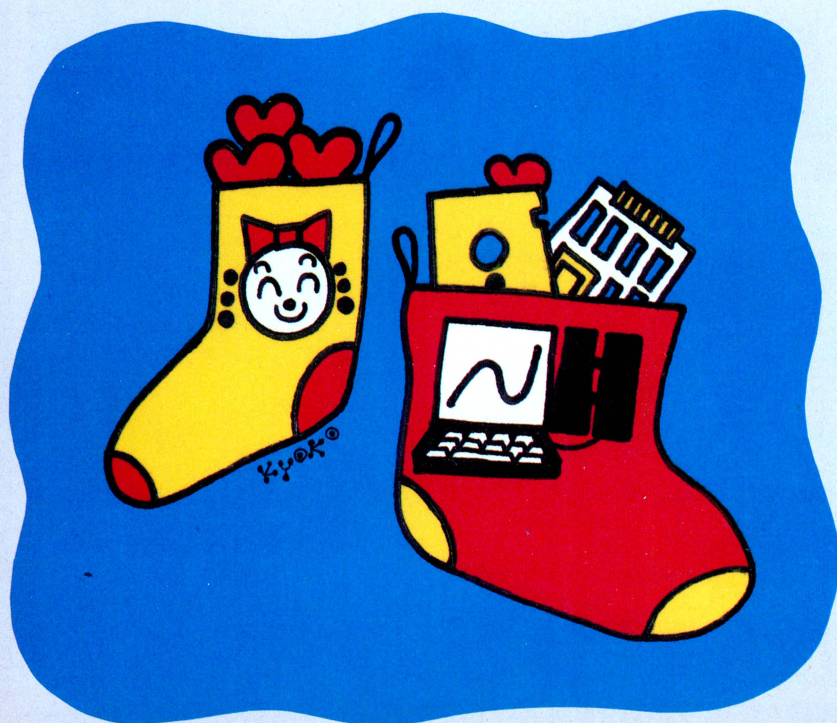
マンションなどによく見られる，ベージュのスチール製の扉です。するとここはマンションなのかしら。ん！ 隣にもずっと扉がある。試してみよう。また，まえに立ちます。

「どうぞ」

やはり声がしたので，ノブを回して開けました。同じように全体が真っ白で，こんどは黒い線が無数に散らばっています。入ろうとすると，外に出てしまいました。1番目の扉の内側にも2番目の扉の内側にも空間というものが存在しないのです。点と線だけの閉じた場所。じゃ3番目はきっと違うかもしれない。

声を待たずに3番目の扉を開けました。すべてがゆがんでいました。部屋なのですがゆらゆらとゆれています。目がおかしくなったのかな。入りかけると，自分の足がゆがむのが見えました。やっぱり目のせいじゃない。怖くなって足をひっこめて扉をバタンと閉じました。

4番目の扉のまえに立っています。迷っています。でも，扉を開けてつぎに進まなければならない。それがこの世界のルールなのだから。開けました。肉が腐ったようなひどい悪臭がぷんと鼻をつきます。薄暗い部屋。天井から何かがぶらさが



時が満ちたわけでもない
力を尽くしたわけでもない
ただ
それが
やってきたのだ



っている。目が暗闇に馴れてきました。天井の梁にロープがくくりつけられていて，ぶらさがっているのは私。自殺した私の姿でした。何日も経っているのか，肉が腐って剥れ落ち白い骨が見えています。白骨。私。腐った体。ますます怖くなっておおいそぎで扉を閉めました。やれやれ。

5番目の扉。こんどは，

「どうぞ」

という声がしました。ゆっくりと開ける。空っぽの四角い部屋でした。中に入って扉を閉じました。天井と床，四方の壁にひとつずつ同じ形の扉がついています。いま入ってきた扉を除けば，すべて鍵がかかっていて開きません。呪文も道具も何もありません。

「しばらく待て」

また，声がいいました。じっと待ちました。時計を持っていなかったので，何時間経ったのか何日経ったのかわかりません。ただ，ぼーっと待つばかり。ふいに，6つの扉がぱっと開き，白い光がすべての方向から強く差し込んできました。体は宙に浮き，くるくるくるくるといつまでも回り続けるのでした。

作品のアイデアは突然にやってきます。まるで，思いがけない人から予期せぬプレゼントをもらうかのように。もっと正確にいうならば，アイデアは心の奥底にいくつも眠っていて，きまぐれに心の表層に浮かび上がってくるといったほうがよいでしょう。私にできることは，ただ待つことだけ。これがなかなかむずかしい。なにもしない，ということが妙にむずかしいのです。

SOFTWARE INFORMATION

年末には超ビックタイトル,「出たな!! ツインビー」が発売されますが，ほかにも「ジェノサイド2」や「大戦略III'90」などの大物が登場しそうです。そして，年が明けての注目作はずばり「レミングス」ではないでしょうか。内容は次号で紹介しますが，なかなかハマりますよ。

大戦略III'90

ウォーシミュレーションゲームの老舗といえばシステムソフト。そのシステムソフトの最新作，大戦略III'90の発売が間近に迫った。

名作ソフト，大戦略シリーズの最新バージョンというだけあって，戦略ゲームの決定版ともいえる内容に仕上がっている。

X68000ではキャンペーン版大戦略IIのつぎに，いきなりこの大戦略III'90が出るわけであるが，PC-9801版ではただのIIIというのがあった。IIIはリアルタイムで進行するうえ，システムもかなり複雑だったので，難易度は高かった。しかし，'90がついたときに命令数が減り，それぞれのユニットに細かい命令を出さなくてもすむようになったので，比較的操作が楽になった。

X68000版では画面がウィンドウっぽいつくりになっていて，レイアウトも自由に変えられる。

X68000用 5"2HD版　9,800円(税別)
システムソフト　☎092(752)5278



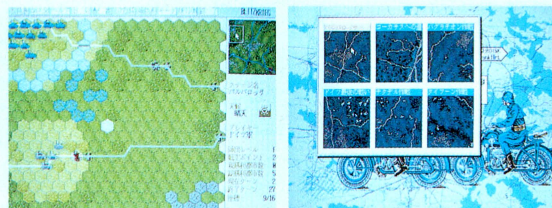
ブリッツクリーク

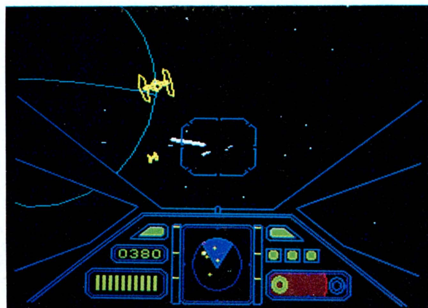
「大戦略III'90」は来年初頭に発売の予定だが，その前にもう1本，システムソフトから本格派ウォーシミュレーションゲームが発売される。名前は「ブリッツクリーク」つまり，“電撃戦”となっている。内容は1941年から1945年までの東部戦線を舞台に，史実に沿った有名な作戦，バルバロッサ電撃作戦やスターリングラード攻防戦，クルクス戦車戦などの1場面をシナリオにしたゲームである。

プレイヤーはドイツの戦車部隊を指揮する機甲師団長を演じ，III号J型やIV号H型などを率いてソ連軍と戦って，制限時間内に勝利目標を確保しなければならない。

ひとつのマップをクリアすると，新たな命令が下されるが，このとき部隊の被害，経験値はそのまま引き継がれる。しかし，シナリオを進めていくと，パンサーG型やタイガーIなどの新型車両も補充されたりもするのでご安心を。

X68000用 5"2HD版2枚組　9,800円(税別)
システムソフト　☎092(752)5278





年末の話題はM.N.M.対コナミで決まり

1.	スターウォーズ	2↑
2.	キャメルトライ	10↑
3.	出たな!! ツインビー	―初
4.	アクアレス	10↑
5.	生中継68	1↓
6.	パロディウスだ!	3↓
7.	ボナンザブラザーズ	5↓
8.	ファランクス	6↓
9.	イース	4↓
10.	パワーモンガー	―初

「パロディウスだ!」が下降傾向になってから仁義なき戦いが続いていましたが，ここにきて下馬評どおりスターウォーズがトップにつきました。発売前からこの強さはすごい。しかし，背後には同じく発売前なのに3位につけている「出たな!! ツインビー」の不気味な姿が。これから年末にかけての見どころはズバリこの2本の対決。こりゃ両方とも4万本は売れなきゃおかしいな（大ウソ）。

このビッグタイトルの間をぬって,「キャメルトライ」と「アクアレス」が健闘しているのが目をひきます。「キャメルトライ」は“タイムトライアルが熱い”“あの回転はすごい”“パドルとかのおまけがうれしい”,「アクアレス」では“動きの感覚が面白い”“面白いソフトを作ろうという前向きの姿勢がよい”など，ゲーム性以外の部分，ソフトハウスの頑張りに対しても評価が高まっているようです。

あとはちょっとどれも降下傾向で元気がないですね。でも,「パロディウスだ!」なんか「生中継68」の前の作品なのに，6位をキープしてるんだから立派といえば立派。あわせて3作品をランクインさせているコナミの元気さが目立ちます。

「パワーモンガー」が初登場で10位。リアルさと画面のつくりのよさ，11月号のレビューの影響などが理由に挙がっていますが，ややマニアックなだけに爆発的な人気ではなく，むしろ息の長い，演歌のような動きを見せるのではないでしょうか。

今後は「ジェノサイド2」,「シムアース」と「飛翔鮫」,それから「フェアリーランドストーリー」あたりも登場して面白くなりそう。では来月まで，アディオス，アミーゴ。（浦）

X68000の日本語FEP「FIXER 4」のバグ修正がPC-VANのX1-clubで少しずつ配布されています。ただし，あくまでも途中版なので使用は自己責任で，質問等もX1-club内でのみお願いします，ということです。通信手段のない人にはバージョンアップのときにサポートされるので，ユーザー登録ハガキを必ず出しておきましょう。

ラストバタリオン

PCエンジンの極めてごく一部に好評のシューティングゲーム「オーバーライド」がX68000用にリメイクされて登場した。自機はホバークラフトで，操作はレバーにショットとスピードチェンジだ。様々なシューティングのおいしいところを持ってきているといえるだろう。目標は狂った惑星上のコンピュータで，舞台は惑星の周囲の宇宙空間から惑星内部へと展開していく。パワーアップはメインショットだけでなく，4種類の形態のビットが2系統あり，ショットボタンを一定時間押してから離すことで強力な波動砲攻撃も可能になる。シンプルに楽しめる純粋なタイプのシューティングゲームだろう。

PCエンジン版とはステージの構成やキャラ等が変更されていて，デカキャラ等も登場する。迫力もアップしているので，別物と考えたほうがいいかもしれない。（八）

X68000用 5"2HD版2枚組　8,800円（税別）
スティング　☎03(3838)0433





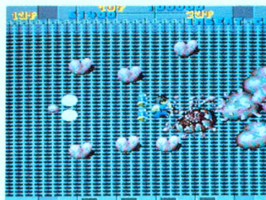
エイリアンシンドローム

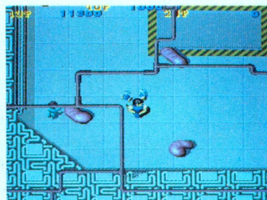
電波新聞社から「エイリアン・シンドローム」が発売されるということは，かなり以前からいわれていた。が，なかなか発売されず，その間に「イース」「キャメルトライ」などが先に発売され，安否が気遣われていた。しかし，とりあえず「キャメルトライ」の次はこのゲームの番のようだ。

このゲームはもともとアーケードゲームだが，発表は1987年ともう4年も前のこととなる。

エイリアンを倒し，捕虜を助けていくという内容からも想像できるように，グラフィックやサウンドがなかなかエゲツない。エイリアンを撃つと「ブシュ」とかいう音とともに破裂する。こちらやられると「キャー」という悲鳴が上がり，さながらホラー映画（スプラッタかな？）のような趣。ぎゃー，ぶちゅぶちゅ。

X68000用 5"2HD版　価格未定
電波新聞社　☎03(3445)8201

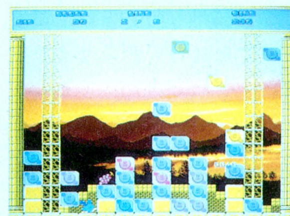




PITAPAT

フィルインカフェというソフトハウスの企画，開発による対戦型アクションパズルゲームがビクター音楽産業から来春に発売されることになった。「PITくん」「PATちゃん」と名付けられたウサギのようなキャラクターでうまくブロックを動かし，消していく。ブロックはタテ，ヨコ，ナナメのいずれかに4つ以上並ぶと消える。ブロックの消え方はコラムスを想像してもらえばいいかな。キャラクターは1ブロック分のジャンプと，ブロックを押すことしかできないので，上から降ってくるブロックを消すのはなかなか苦労する。マップは全64面で，途中には何枚かのCGも隠されている。画面や音にも力が入っているし，燃えそうな対戦モードもあるので，期待してもいいのでは？

X68000用 5"2HD版2枚組　6,800円（税別）
ビクター音楽産業　☎03(3423)7901



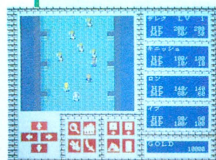


ヴェルスナーグ戦乱

かつてヴェルスナーグに存在し，繁栄の頂点に達していた「魔法王国」は，突然現れた「魔竜」によって崩壊した。「魔竜」とヴェルスナーグを治める「神」との戦いは数百日にもおよんだが，神が自らの命と引き換えに「魔竜」を封印することで終止符が打たれたのである。しかし，そのとき世界の大半が破壊された……。それから2000年以上がたち，ヴェルスナーグには新しい文明が生まれつつあったが，世は戦乱の混沌のなかにあった。

ファミリーソフトからX68000のオリジナルRPGが発売される。上のような世界を舞台に，全20のシナリオで構成されたキャンペーンが進行。パーティーを組んで旅をする，というわりと普通のRPGだが，隊列の組み方や戦闘方法などに工夫がなされている。

X68000用 5"2HD版　価格未定
ファミリーソフト　☎03(3924)5727





プロサッカー68大会

イマジニアからスポーツゲーム，「プロサッカー68」が11月29日に発売されますが，それに先立って，パソコン雑誌編集部対抗の「プロサッカー68」大会が10月3日に行われました。

優勝賞品はシャープ提供の4インチカラー液晶テレビ「CRYSTALTRON」。試合は5分ハーフで争い，時間内で勝負がつかない場合はPK戦で決着をつけるというルールでした。3位以下の順位もPK戦で決められました。

で，Oh!X代表（私，つまり編集Aなのですが）はとんとんとんと勝ち抜いていき，まさかまさかの優勝に輝きました。あまり（というか，ほとんど）練習をしていなくて，1回戦でログイン代表の忍者増田君と当たったときに，「ログインなら負けても文句いわれないな」などと思っていたぐらいだったんですがね。しかし，勝ったからいうわけではないのですが，この「プロサッカー68」の対戦はなかなかに面白いものでした。物がかかっていたからともいえますが，さすがに決勝では燃えました。

私も含めて，参加者が全体的にまだゲームに慣れてないという感じだったので，もっと練習してから再度対戦してみたい気もします。

結果は以下のとおりですが，詳しい途中経過は174ページのmicroOdysseyをご覧ください。

チーム	順位
テクノポリス	3
ポプコム	5
コンプティーク	2
マイコンBASICマガジン	6
マイコン	4
Oh！X	1（優勝）
ログイン	7





私は町のケーキ屋さん

Kaneko Shunichi
金子 俊一

フェアリーランドの片隅にあるアルアルファ王国。その国の王様がドラコリスクという大トカゲの魔力で伝染病にかかってしまった。で，そのひとり娘トレミーと友達のロドミーがドラコリスクを倒すために旅に出たとさ。

タイトーのキャラクターゲーム,「フェアリーランドストーリー」が届きました。タイトーといえば,「ちゃっくんぽっぷ」「バブルボブル」などの発売元で，キャラクターゲームの大御所ですよね。そして，移植を担当したのはSPSとくれば，完全移植と考えてもいいでしょう。

このゲームは「1985年もの」ということで，ノスタルジックな味わいがあります。ひらたくいえば，ちょいと古いってとこかな。

必殺！　キャラゲー

そもそもキャラクターゲームとは，ゲームのシステムなんかより，登場キャラに力が入っているものだと思う。開発側のキャラへの思い込みが感じられるゲームこそが，キャラクターゲームといえるのではないだろうか。

この「フェアリーランドストーリー」も例外ではなく，キャラはかわいい。バブルボブルより1年早い登場ということで見た目はちょいと地味っぽいが，それなりの味わいがあるのだ。

主人公はアワ吐き怪獣ではなく，魔法使いの女の子だし，ブタはジャンプするし，リボンをつけた恐竜は火をふく。ああ，メルヘンの世界（とは程遠い)。

また，キャラクターゲームの特徴として，見た目のかわいさと裏腹に，プレイのむずかしさという点があると思う。

キャラに惚れ込みすぎて，テストプレイに気合いが入りまくっているためじゃないかと邪推してしまうくらいだ。そして，後述するように，このゲームもむずかしい。

構成や，ゲームデザインは「バブルボブル」の元ネタともいえるくらいに似ている。これは期待しててもいいだろう。

バトンをくるり

このゲームの基本を伝授しよう。自分で使える技はたったひとつ。スティックを振るだけ。それだけじゃちょいと弱そうでしょ。ところがどっこい，そこから出る魔法のビームは敵をケーキにしちゃうのだ。

某ゲームならそのケーキは食べちゃうとこだろうけど，このゲームでは破壊するだけ。ケーキを下に落として壊すのが基本ワザ。もしケーキの下に敵がいれば，その敵も消える。そうやって，いっぺんにたくさんやっつけると，どんどん高得点になっていく。

それじゃ画面のいちばん下にいるときはどうすんだって？　心配ご無用。敵がケーキになっても気にせずに，ビームをいっぱい当てると，ケーキは消化されちゃうのだ。ああもったいない。食べ物は大切にしようね。

画面にいる敵を全部やっつけると1面クリア。全部で100＋1面で構成されている。全部の面が凝りまくっているわけではないのがちょっと残念。

簡単な面は“ちょー楽勝”だが，ムズい面は“死んでも解けない”。このギャップが激しいのは愛嬌のうちだと思うしかないだろうね。80面あたりから厳しさが増して，90面あたりからはただの拷問といってもいいぐらい。

味方はカレーの王子様

それでもくじけちゃいけない。アイテムという強力な味方がいるのだ。お月さまは敵を全部ケーキにしちゃうし，お星さまは当てるとイタイ。面飛ばしやファイヤーもある。ほとんどのアイテムは強力だけど，難点は有効期間が短いこと。いい気になってると悲しい目にあってしまう。

教訓：驕れるものは久しからず

面によっては仕掛けもある。ワープドアと虫の舌だ。ワープドアはその名のとおり。問題なのが虫の舌。ワープドアから芋虫が出てきて，カメレオンよろしく舌をビェーと伸ばすのだ。もちろん，つかまっちゃい



頭の上に乗っても大丈夫



ビームで敵をケーキに変える



黒コゲになって死ス



X68000用 5"2HD版2枚組　6,800円(税別)
SPS　☎0245(45)5777

けない。
「君子危うきに近寄らず」といきたいところだが，敵が芋虫につかまると7000点ももらえる。得点が低めのこのゲームでは，かなり魅力的なのは否定できない。「虎穴にいらずんば虎児を得ず」で，ついつい芋虫の前で追いかけっこをしてしまう。グァッデーム，またつかまっちまったい。きっと君も繰り返すぞ。まさにおいしさと背中合わせの河豚の毒。
昔，dBソフトから発売されていた「ラプテック」というゲームで，カメレオンが舌をのばしてキャラクターを捕まえていたが，似たような趣向があるぞ。

欲張っちゃあ負けよ◆◆◆◆◆◆◆◆

バランスがいい意味で悪いので，スコアメイクに走らず，面クリだけを主体に考えていれば，比較的簡単に90面くらいまで行ってしまうだろう。
ただし，スコアメイクを考えると完全にドツボの世界が待っている。一度にたくさんの敵をやっつけるのが高得点の基本なので，ぎりぎりまで敵を引きつけなければならないのだ。追いかけっこの要素が強く，敵キャラのアルゴリズムを読み取って誘導しなければならない。引きつけるという行為ゆえ，敵が必ず近くにいるので，ちょっとしたミスが命取りになってしまうのだ。何度ジョイスティックを破壊しようと思ったことか。ああ，自己嫌悪。
その分，成功したときの快感は大きい。この味を知ってしまった日からが真の「フェアリーランドストーリー」の始まりといっても過言ではないだろう。
こだわり始めると，１面ごとに攻略法を研究してしまう。もちろん，対象は“ハイスコアでのクリアの手順”にほかならない。ところが，これといった決め手を見つけるのが大変なのだ。「バブルボブル」などよりランダム性が強い気がする。



WORMに敵がつかまると高得点



普通に死ぬと泡になる



あっ，コイン見っけ



WORMに気をつけろ！

戦いすんで日が暮れて◆◆◆◆◆◆◆

やっとの思いでクリアすると，メッセージにスペルミスがある。きっと完全移植のなせる技なのだろう。どこにあるのかは自分の目で確かめてもらいたい。
ところで，このゲームは２人同時プレイではなく，１ライフの交代プレイになっている。その２プレイでのコンティニューの仕様が（もともとがそうなのであろうが）少し気になる。
具体的な話にしよう。
もし両方のプレイヤーにextendがなかったら，交代プレイなので２プレイヤーが最後にプレイすることになる。１プレイヤーが10面，２プレイヤーは４面でゲームオーバーになったとすると，ラウンドセレクトは４面までになってしまう。ここで１プレイヤーにextendがあったとすれば，最後にプレイするのは１プレイヤーになり，１プレイヤーが死んだ面（つまり10面以降）からセレクトできることになる。
ようするに，「最後にプレイした人が進んだ面までセレクトできる」ようになっているのだ。極端な話，１プレイヤーがノーミスでクリア，２プレイヤーは１面で全滅した場合では，ラウンドセレクトは１面からになってしまう。やはり，面が進んでいるほうか遅れているほうに合わせてセレクトできるように統一を図りたいところ。
また，なにかと比較をするようで申し訳ないのだが，X68000用の「フェアリーランドストーリー」と「バブルボブル」ではジャンプボタンと攻撃ボタンが逆になっている。プレイしにくいし，間違えやすい。せめてセレクトできる程度の心遣いがほしかった。会社が違うとはいえ，後発が努力をすべきである。

本当の姿◆◆◆◆◆◆◆◆◆◆◆◆◆

このゲームをやってから思った。バブルボブルは上達すればするほどに一方的な攻撃になっていった。有無をいわさずアワに閉じ込め，一気に全滅させるのが快感だった。あれは「サディズム」ともいえるゲームではないだろうか。このことはサイバブルンでもそうだった。
一見「バブルボブル」と似たように見える「フェアリーランドストーリー」だが，根本的に違うのだ。うまくなるほどに自分へのかせが増えていく。耐えて耐えて耐え抜いて，快感はそれを成し遂げたこと。これはまぎれもなく「マゾヒズム」のゲームだったのだ。
真のゲーマーなら自分への挑戦という意味でも「フェアリーランドストーリー」に挑んでもらいたい。このゲームをノーコンティニューでクリアできる人を見てみたい気がする。

乞うレインボーアイランド

正直にいおう。タイトーにとって１年の差は大きかったようだ。システム，グラフィック，ミュージック，その他，トータル的に「バブルボブル」のほうが素晴らしい。ゲーム性が高いのだ。両方とも遊ぶことができるX68000ユーザーは幸せというべきなのだろう。どちらも完全移植なので，「フェアリーランドストーリー」はちょっと分が悪い。あとは趣味の問題なのでなんともいえないが，値段が安めなのはOKではないだろうか。
次はやはりどこからか「レインボーアイランド」が出るのを待ちたい。

総合評価（0　5　10）

システム	★★★★★★★
グラフィック	★★★★★★★
ミュージック	★★★★★★★★
サウンドエフェクト	★★★★★★★★
快感	★★★★★★★★
苦行	★★★★★

スピード感あふれる本格派

Kageyama Hiroaki
影山 裕昭

日本でも人気を高め，国民的なスポーツのひとつとなりつつあるサッカー。それを本格的に，真面目にゲーム化したのがこの「プロサッカー68」。野球ゲームは根強い人気があるけど，サッカーゲームはいかがなもんなんでしょうか。



つい最近「パワーモンガー」を発売したイマジニアから，またまた海外の話題作が発売されることになった。それもいままでとはうってかわってのスポーツゲーム，その名も「プロサッカー68」。

すでにスーパーファミコンでも,「プロサッカー」のタイトルで発売されているようなので，プレイした人もいるかもしれない。原作は「KICK OFF」といい，ヨーロッパで大ヒットしたゲームだそうだ。

海外の話題作っていうと，日本のソフトハウスなら無視するような細かい部分にも凝っていて，プログラマのゲームに対するただならぬ思い入れが，ひしひしと伝わってくることが多いように感じる。

さて，この「プロサッカー68」ではヨーロッパの国民的人気スポーツであるサッカーを，どのようにディスプレイ上に再現しているのだろうか。

ハンパな気持ちじゃだめ◆◆◆◆◆◆◆

やってみてすぐに思ったことは，ボールと選手の動きが異常に速いということである。おまけにボールコントロールが難しくて，思った方向にボールを蹴ることはおろか，ドリブルさえ満足にできないのである。普通，サッカーゲームというと，ボールを持った選手を動かすとドリブル，止まるとボールキープするようになっていると思っていた。

ところが，このゲームではドリブルとボールキープは明確に区別されていて，ボールを持った選手を動かすとドリブルはするが，止まるとボールはキープされず，そのまま蹴っていた方向に転がっていってしまうのだ。ボールキープするには，選手とボールが離れている状態でAボタンを押さなければならない。そして，Aボタンを押し続けている間，選手はボールキープを続けるが，この状態からドリブルに移ることはできない。

ここがミソだ。Aボタンを離して味方にパスするしかないのである。パスの方向は方向キーで決めることができるのだが，味方のいる方向を止まってから確認しているようでは，相手の選手にボールを取られてしまうことが多くて油断できない。常に自分以外の選手のいる位置を把握しておくことが大切である。

こう書くと簡単そうだけど，操作だけでも大変なのに，ほかの選手に気を配るなんて本当に大変なことなのだ。

さらにさらに，ドリブルからボールキープに移る動作も難しい。つまり，ボールから離れた状態でボタンを押せばボールキープだが，タイミングを間違えて選手に触れたところでボタンを押すと，止まるどころかシュートになってしまうのだ！ ドリブルの間，選手とボールが離れている時間はコンマ何秒かである。この時間を見極めてボタンを押さなければいけないのだから，



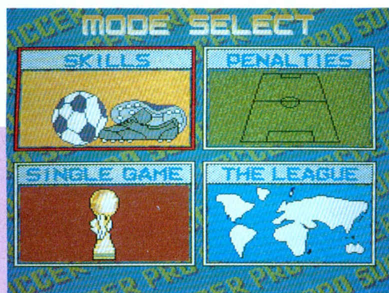
チームは有名どころが

いかにボールコントロールが難しいかわかってもらえるだろう。

さらにさらにさらに，やればわかるが，ドリブルもボールを左足よりに蹴るか，右足よりに蹴るかによって，方向転換のしやすさが変わってくる。コンピュータはくるっと180度方向を変えてドリブルを続けていたりもするが，僕にとっちゃ直角に曲がるのでさえ難しいのに，Uターンするなんて至難の業である。

以上のようにボールコントロールに関しては非常に高度な技術が要求され，本物志向と呼ぶにふさわしい出来である。上達するには時間がかかるだろうけど，それだけに腕の違いがもろに出るわけで，友達を相手にプレイして高圧な態度に出ることもできよう。ただし，あんまりにも相手をけなして，夜中に無言電話がかかってきても知らないからね。

さて，これまで選手の操作方法について



X68000用 5"2HD版 9,800円(税別)
イマジニア ☎03(3343)8911



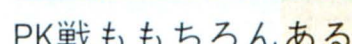
シュート！ キーパー飛びつくが

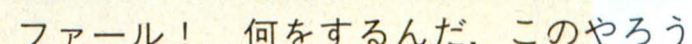


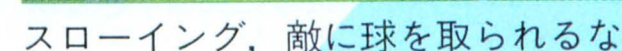
ゴールされてしまった

書いてきたが，そろそろゲーム全体の紹介をしよう。画面写真を見てのとおり，試合中はグラウンドの一部が表示され，ボールの動いた方向に合わせてグラウンドも全方向に高速スクロールするようになっている。また，選手やボールの位置は，画面左上にあるグラウンド全体図で確認できるようになっている。上下にそれぞれのゴールがあって，ハーフタイムでは実際のサッカーと同じく攻める方向が逆になる。つまり前半上攻めなら，後半は下攻めになるわけだ。多くの人は下攻めになるとやりづらくなるだろうから，前半を上攻めにして攻めまくり，後半は守りにまわるといった作戦も立てられる。

先ほども触れたように，選手とボールの動きはゴキブリの素早さを連想させる速さで，「速すぎる～」って感じ。試合中はボールに近い選手にプレイヤーが直接操作できる目印として，三角のカーソルがつく。このカーソルの向きによって，攻める方向がわかるようになっていることも親切でうれしい。キャラクターのアニメーションにもぎこちない部分がない。さすが海外でヒットしただけのことはある。ビジュアル面については，スポーツゲームのもつスピード感とスリルを味わうことのできるゲームスピードといい，選手の動きといい実に素晴らしい出来で感心した。が，よくできているだけに，ボールが白のベタ塗りであったことが気になった。効果音もボールを蹴る音と歓声しか入っていなく，ちょっと寂しい感じがした。


PK戦ももちろんある


ファール！　何をするんだ，このやろう


スローイング，敵に球を取られるな

シュートが決まり，喜んで宙返りする

コーナーは慎重に

細かい設定がうれしい◆◆◆◆◆◆◆◆

選手の操作が難しいために，このゲームには練習モードが用意されている。まずはここで選手の操作に慣れよう。ペナルティーキックの練習ができるモードもある。

実戦モードには1試合限りのシングルゲームと，8カ国を相手に総当たり戦で争う“THE LEAGUE”が用意されている。シングルでは友達を相手にして対戦することもできるし，コンピュータを相手にすることもできる。選手のレベルは5段階から決めることができ，動くスピードなどが変わってくる。まずは“SUNDAY LEAGUE”でゲームに慣れて，操作に慣れたらどんどんレベルを上げていって，最終的には“INTERNATIONAL”でプレイできるようになろう。

では，試合。まず，フォーメーションを決める。フォーメーション，つまり攻撃を優先するか，守備を優先するかで，いくつかのフォーメーションシステムが選べるのである。プレイヤーは4種類のフォーメーションシステムからひとつを選択する。

最も一般的なのが4-3-3システム。つまり，フォワードとミッドフィルダーが3人ずつ，フルバックが4人のシステムである。普通に使うなら4-3-3，攻撃重視なら4-4-2，守り重視なら5-3-2，攻撃も守りもこなしたい場合には4-2-4を使うといいだろう。

ハーフタイムでもフォーメーションの変更ができるので，前半を4-4-2システムで積極的に攻撃して点数を稼ぎ，後半は5-3-2システムで守りを固める，といった作戦を立てて遊んでみよう。

ハーフタイムの設定も5分から45分まで5種類の中から選ぶことができる。45分ハーフを選ぶと，フルタイムで90分。うーん，なんだか体力使いそうだな～。試合中は風も吹くので，ボールの動きが思わぬ方向に曲がる場面も見られる。ほんとにリアルに作ってあるようだ。

やはり，本場ヨーロッパのサッカーゲームということで，翼君のような派手で豪快なシュートは打てないけど，選手の操作系に工夫を凝らして，最も基本的なドリブル，ボールキープを難しくしたところが，さすが本格的といったところか。フォーメーションシステムの再現や，グラウンドに風まで吹かせたりするところにも，こだわりが見えるよね。派手な演出は少ないけど，やればやるほど味が出てきてやめられない，まるで酢昆布のようなゲームだ。

目指すは芸術的プレイ

本文では触れなかったが，このゲームでは相手選手に近いところでAボタンを押すと，タックルやスライディングをすることができる。しかし，本物のサッカーと同じようにイエローカード，レッドカードがあって退場させられてしまうこともあるから，多用は危険，使うのは最小限にしておきたい。

さて，コンピュータのプレイを見ていると，実に鮮やかにパスを回しているが，人間であそこまで美しくプレイできる人がいるのだろうか。上手になって再現プレイなんかあれば，とても楽しめたと思うんだけどな。いまは，基本の三角パスさえできない僕だけど，大会で優勝した（A）氏にでも特訓してもらうか。

総合評価	0　　5　　10
スピード感	★★★★★★★★★
グラフィック	★★★★★★★
操作性	★★★★★★
起動時間の速さ	★★★★★★★★★
熱中度	★★★★★★★

掌中の機動戦士たち

Ishigami Tatuya
石上 達也

機動戦士ガンダムのキャラクターで戦略ゲームが楽しめる。ファンにとってはたまらない喜びだろう。あのモビルスーツをこうやって，こいつはこういう性能だからそっちにぶつけて……。想像力は膨らむばかりですな。



プチ（フィルムを繋ぎ合わせた跡の音）
しばしの間。
ズンチャン，ズンチャン（ドラムの音）
も，え，あ，が，れ，（Cのコードで）
もえあがーれ（Dm7のコードで）
⋮

いいなあ，「機動戦士ガンダム」。いままさに，X68000であの感動を再現できるときがやってきたのです。と，期待に胸をふくらませながら，いざ，ゲームスタート。

ゲームの概略

このゲームは，基本的にシミュレーションゲームです。が，三國志などのように際限なく手間暇かかるゲームではありません。どちらかというと「銀河英雄伝説」のような感じで，気軽に立ち上げて1～2時間でひとつのシナリオを終了させることができます。また，じっくりと取り組みたいという方には，5つのシナリオからなるキャンペーンモードも用意されています。しかし，キャンペーンモードではプレイヤーは連邦軍しか選べませんので，密かにジオン軍のメカに心惹かれている人は，ちょっとがっかりかもしれません。

ゲームのスタイルは，先攻後攻を選び，自分のターンごとに自分のキャラクターに移動，攻撃を指令し，相手の旗艦を沈めるというものです。とはいっても，相手の攻撃ターン中でも自分のキャラクターは適当に応戦するので，「先者有利則」が必要以上に働いて，非常に悔しい思いをすることはありません。また，旗艦はストーリーごとに固定で，生き残った艦へのたらい回し指名はできないので，打ち逃がしたたった1隻の巡洋艦を追って，右往左往するようなこともありません。

画面写真からわかるように，キャラクターなどの移動には基本的にHEXの概念を用います（しくしく）。



X68000用　5"2HD版3枚組　9,800円(税別)
ファミリーソフト　☎03(3924)5727

バックストーリー

タイトルからもわかるように，題材はあの「機動戦士ガンダム」です。私を機械工学科に行かせしめた「機動戦士Ζガンダム」の前作です（最近は専攻を尋ねられると「98式イングラムのマニュピレータにチョウチョ結び機構を付加しています」と答えるようにしている）。

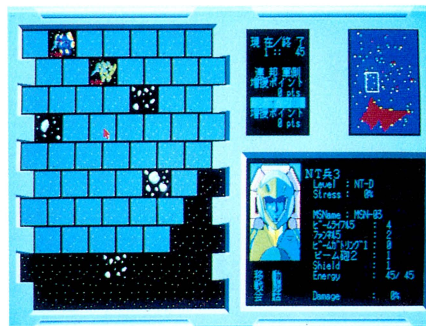
で，そのクラシック・オペレーションなわけで，宇宙暦0079年あたりの話です。セイラさんやスレッガー中尉が乗っている機体がコアブースターであることや，マニュアルにはガンタンクが載っていながらもハヤトはガンキャノンに乗っていたりというところから見て，テレビシリーズではなく，映画のほうのストーリーのようです。

また，ドライブ1に入れるディスクをディスクBからディスクCに切り替えることで，それから14年後の映画「復活のシャア」のストーリーを遊ぶことができます。しかし，どのストーリーを選んでも，タイトル画面は地球を背景にラー・カイラム（ひょっとしたら，居住スペースをなくしたネール・アーガマ？）なのはご愛敬です（でも，なにか神経を逆撫でられるようなものがある）。

と，ここまで読んできてお気づきかもしれませんが，バックストーリーを知らないとこのゲームはちょっとわかりづらいものがあります。レウルーラを沈めたときの快感や，手持ちのキャラがジムIIIばかりのときに，ジェガンをやられた悔しさがわからないのでは，このゲームの面白さが半減してしまいます。

一応，マニュアルには，ひととおりのキャラクターの説明が載ってはいますが，あくまでひととおりでしかありません。ガンダムの世界では，時代とともにメカの新旧交代が起こっていて，向かうところ敵なしの最強メカが次のシナリオでは量産されていたりと，そのシナリオの時点でのメカの相対的な強さや普及率などが違ってきます。このようなデータはマニュアルからでは読み取れません。

「提督の決断」のように，やられたときにグラフィックが出てきたり，やられたメカの重要さによって，より悲しい音楽が鳴ったりするとかして，「ガンダム」への感情移入を手伝ってくれれば，もう少し，一般ピープルも楽しめたと思うのですが。



移動可能な場所はひと目でわかる



戦闘シーンでは荷粒子砲や弾が飛び交う

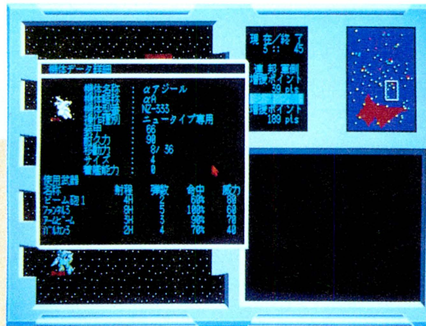
キャラクターが見えづらい◆◆◆◆◆◆

その昔，ラ・ポートという出版社からPC-8801やFM-7用に「機動戦士ガンダム」というゲームが発売されていました。徹底的に見せるゲームで，CRT上には次から次へと，映画やテレビでのシーンが描かれていきました（例によって，ゲームの内容はほとんどない）。

当時のメディアといえば，半ば常識的にテープだったのですが，そのロードとロードの間には，本物の声優さんによるセリフや効果音が入っていたのでした。当時の技術でできるところまでやってみたというようなソフトでした。ただし，このようなソフトの常として，マニア以外には受けなかったのではないかとは思いますが。

さて，やはりガンダムを題材にするからには，そこまで要求してしまうのはファンの当然な心理なわけで，そういう点から見るとこのクラシック・オペレーションには，やや不満が残ってしまいます。「機動戦士ガンダム」の魅力であるキャラクターの個性があまり生きていないのです。また，上級兵士1とかニュータイプ兵2とかいうのもいただけません。せめて，ザビ家の人間ぐらいは本名で登場させてほしいものです。ときどき，将棋の桂馬や飛車にザクとかゲルググとかいうシールを貼って，プレイしているような気になったりします。

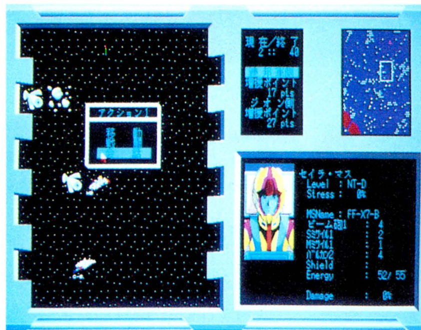
たとえば，ガンダムがホワイトベースから発進するとき，



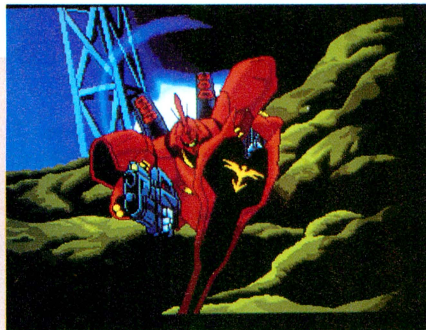
ユニットごとのパラメータも豊富



会話をすると相手のステータスがわかる



もちろんセイラさんもいるぞ



戦いを終えると，このような画面が

セイラ「アムロ，発進よろしくて？」
アムロ「はい，セイラさん」
（ガンダムがカタパルトに乗る音）
アムロ「アムロ，ガンダム，行きまーす」

などと，PCMで，とまではいかなくても，画面にセリフを出すなどして，しゃべるようなことをすると盛り上がったのではないかと思います（当然，このような一般人にはうざったい機能は，スイッチで禁止できるようにしておく必要はある）。

ただし，同じガンキャノンでも，ハヤトとカイが乗るのとでは，弾の命中率や被弾しやすさに違いがあったりとか，細かいところでキャラクターを考慮しているようなところも若干ながらあります。

操作性◆◆◆◆◆◆◆◆◆◆◆◆◆◆

やはりゲーム時間が1時間前後とはいっても，それなりの時間このゲームに触れているわけで，ちょっとのことでも気になってしまいます。特に，拡大する領域を右側の全体地図から選んで，左側のウィンドウにもってくるとき，その領域の指定に改良の余地おおありです。

また，敵のキャラクターの種類（たいていアルファベットの2文字でグラフィックの下にちょこっと書いてあります）が15インチのCRTではちょっと見づらいですし，ダメージなどもそのキャラクターのとなりに自分のキャラクターを寄せて「会話」をしなければわかりません。

直接操作性には関わってきませんが，コンピュータの攻撃ターンのとき，画面が目まぐるしく変化していって，何が起こったのかさっぱりわかりません。キャラクターの移動ぐらいアニメーション処理してくれてもいいと思うのですが。

対戦プレイも◆◆◆◆◆◆◆◆◆◆◆◆◆

さて，このクラシック・オペレーションのように，思考型（というのもなんだが）ゲームのもうひとつの楽しみ方に友達を家に連れてきて，2人で対戦を行うというものがあります。相手が，コンピュータではなく人間であると，いままで使っていた技が使えなくなったり，予想もできなかったような戦法で攻撃されたりと，新しい遊び方ができるものです。

また，対戦プレイでは，ゲームそのものを楽しむほかに，RPGなどのようにプレイヤー同士のコミュニケーションも楽しいものです。お題が「機動戦士ガンダム」となると，特別な感情がふつふつと体をかけめぐって，あやしげなオタク同士の会話に花が咲くのではないでしょうか（「朱に交われば気持ちいい」）。

画面の端っこに，ちょこっと出てきたメカを見つけては，「いま，岩の向こうに旧ザクが出てきたの見たか？」とか，「見た見た，いま出てきたやつ，TVシリーズのとちょっと胸のあたりの形が違うだろ」などと，昔，騒いでたような連中と対戦すれば，盛り上がること請け合いです。

最後に。「オタクだっていいじゃない」。

ガンダムが好きだから

さらに余計なおせっかいなのですが，このようにオタク会話のお題として設計するのであれば（そんなことを考えて設計してはいないとは思いますが），もうちょっと小技に凝ってもらえればうれしかったです。たとえば，宇宙空間にホビーハイザック（戦闘機能のないスポーツ用のモビルスーツ）がなんの前触れもなくプカプカしているのを，見つけたりすると，オタク心は最高にくすぐられるのですが……。

そうそう，マニュアルは美樹本晴彦さんによる表紙がついています。もっとも，中身は設定資料の超ダイジェスト版ですが……。

総合評価（0〜10）

操作性	★★★★★
ビジュアル	★★★★★★★
サウンド	★★★★★★
熱中度	★★★★★★★★★
ガンダム	★★★★★★★★★★

整ってるね，優等生RPG

Komura Satoshi
古村 聡

国中を怪物たちが襲い始めた。平和な国の動物たちが凶暴な怪物にその姿を変え，ついには徒党をくんで村を襲うようになったのである。この異変の原因を探り，平和を取り戻すために剣士シオンは旅に出た。



ある日，王様から道士ロキテルへの手紙を託された。異変の原因を究明し，力になってほしいと伝えるための手紙だった。手紙を受け取って，ロキテルはこう語った。

「いにしえの災禍という，王家に代々伝わる話がある。昔，ある魔道士がこの世界の征服を企み，破壊の魔神を生み出したそうじゃ。しかし，破壊の魔神は強力でその魔道士にも操ることは不可能であったのじゃ。破滅を待つのみであったこの世界に4人の勇者が現れ，この魔神を倒したのじゃ」

「それでは，魔神を復活させようとしている魔道士が現れたというのですか？」

「そうじゃ。そして，それができる者はこの国ではただひとり……」

そのとき，そこに何者かが現れた。

「お前が道士ロキテルか？ ネクトラル様の命令だ。お前を殺す」

現れたのは，巨大なミノタウルス。僕には何もできなかった。そう，何も……。

「小僧，お前もこうなりたくなかったら，ネクトラル様に歯向かおうなどとは考えぬことだ。ふっ。あばよ」

無念だった。

王にロキテルの死を伝えたあと，ロキテルの孫娘，ルーンとともに旅に出た。ロキテルを殺したミノタウルスやネクトラルを倒すため，そして僕自身のために……。



ぞろぞろと歩き回るフィールド画面

X68000用 5″2HD版3枚組 5,900円(税込)
ブラザー工業(TAKERU) ☎052(824)2493

システムは国民的◆◆◆◆◆◆◆◆◆◆

この「ノーブルマインド」は典型的なロールプレイングゲームであります。キャラクタたちはチップで構成されたフィールドをぞろぞろと歩きまわり，町のチップに触れれば，町の中に入り大勢でぞろぞろと人の間をぬって歩き，また，フィールドやダンジョンを歩いていると，突然，大きなキャラ描画の戦闘モードに入る。……はっきりいえば，まったく国民的ゲーム，つまりドラクエタイプのゲームなのですね。ゲーム中の操作もすべてジョイスティックやジョイカードでできるようになっている（というかそれを前提にしているフシも随所に見受けられる）ので，ジョイカードなどで遊んでいると本当に画面のきれいなファミコンゲームをやっている気分になります。

画面は256×256ドットモードを使っているのですが，色を有効に活用しており粗さがあまり目立ちません。マップも非常に美しい。グラフィックはこのゲームのひとつのウリといえます。画面の描画も素早く，何をやってもレスポンスが小気味よく返ってきます。プログラム的にはかなりしっかりできているのでしょう。

そして，バランスに関しては，これほどよくできているものはあまりないでしょう。ヤバイッと思ったときにはやっぱり死にますが，死にまくってハマるようなことはありません。かといって，やさしすぎるということもなく，ちゃんと楽しめる。おそらく，しっかりテストプレイをした結果なのでしょう。これほどゲームバランスがまとまっているのは，私が見てきたかぎりではおそらく，「イース」以来です。



こいつがにっくきミノタウルス

唯一，このゲームに関して文句をつけるとすればシナリオでしょうか。イマイチ盛り上がりに欠けますね。もう少し，喜怒哀楽がストーリーにあってもいいと思うのですが。名作といわれるようなゲームと比べて劣るとすればそのへんでしょう。

が，とにかくゲームはすべての要素がよくまとまった作品です。間違いなくかなりの秀作である，といえると思います。おしむらくはこれといってとびぬけたセールスポイントに欠けることでしょうか。よくまとまったゲームであるだけにとても残念です。次回作ではそのあたりを考えてもう一歩踏み出せば，秀作を超えて，名作と呼べるものができるのではないでしょうか。

よく炊けたごはん

このゲームは食べ物にたとえれば，本当にうまく炊けた白米のご飯なのだと思う。水加減も米もいいから，固すぎず，べたべたもしてない。つやつやしてて，ほかほかしてる。毎日食べても食べ飽きない。そういうタイプのゲームだ。

しかしながら，それだけあればよいというものではない。なにかが物足りない。そう，おかずがほしいのだ。梅干しだって，さんまの蒲焼きの缶詰でもいい。何か華がほしいのだ。

このゲームは本当にソツなく作ってある。文句のつけようはほとんどない。しかしながら，それだけでは佳作の域を抜け出すことはできないのだ。もうひと息何かがあれば，このゲームはイースなどを超えることができただろう。

総合評価	0 5 10
バランス	★★★★★★★★★
ジョイパッド	★★★★★★★★
サウンド	★★★★★★★
マンドラゴラ	★★★★★★★★
おすすめ度	★★★★★★★★

飛び交う昆虫を撃ち殺せ



PCエンジンで発売されていた「サイバーコア」がX68000に登場。で，内容はというと，ジュラ紀のような世界を舞台に虫を相手に縦スクロールシューティングゲームする，とでもいえばいいのかな。……あなた，虫は好きですか？

Takahashi Tetushi
高橋 哲史

決して私だけが特別敏感だ，ということではないと思う。皆さんにも経験がないだろうか？

そよ風に揺れる木の葉から落ちてきた毛虫の一群に驚いて，自転車のハンドルを切り損ね，そのまま木に激突してさらにどつぼにはまったこと。なにげなく蹴飛ばした石の下が実はムカデの温床になっていて，身の毛のよだつ思いをしたなんてこと。

……なに，ない？ おかしいなあ。そんなはずはないんだけど。なぜこんなことをいいだすのかといいますと，今回の「サイバーコア」をプレイしていて同様の感覚を味わってしまったからなのです。ああ，甦る幼少の頃の記憶。さて，そんなサイバーコアの世界っていったい？

アリアリコロコロ◆◆◆◆◆◆◆◆◆◆

「昆虫シューティング」。このサイバーコアをひと言で表現するとこうなります。敵も昆虫なら自分も昆虫，あまつさえアイテム運んでくるのも太った蜂だったりするので，画面は昆虫一色になります。しかも，形状だけが昆虫昆虫してるのではなく，動きもそれらしく再現されているのですからたまりません。ムカデが這い進むさまなんかもう最高ですよ，奥さん。

基本的にはアイテム奪取パワーアップ型のシューティングになっていますが，パワーアップの方法は少し特殊です。さきほどいったようにファットな蜂くんがパワーアップアイテムを運んできてくれるのですが，それぞれ赤青緑黄と色分けされています。そして，青ばかり取り続けると自機が青の方向に進化していくのです。パワーアップはそれぞれ3段階にわたって行われますが，個人的には緑進化のバリヤーブーメラン（と勝手に呼んでいる）が好きです。



パワーアイテムを落としていく太った蜂？

X68000用 5"2HD版 7,800円(税別)
SPS ☎0245(45)5777

いけいけごーごーっ◆◆◆◆◆◆◆◆◆

さあ，出撃だ！ 全8面を駆け巡るぜ！

1面目はまあ手慣らしといった感じでそれほど難しくありませんが，ボスとの遭遇時に「王蟲(オーム)がイクラを吐いて攻撃してくるっ」と感じてしまいます。

2面はビルと荒野の繰り返しという，よくわからない構成になっています。編隊を組んで（徒党を組んで，かな）やってくる青い蜂さえやり過ごせば比較的楽にクリアできると思います。

3面目。たまに花からパワーアップアイテムが出るので，がっちり取りましょう。

4面は砂漠面です。どうでもいいかもしれませんが，個人的に緑の木の根から出てくるダンゴムシの親戚みたいなのがすごく恐いです。夢に見ちゃいそう。

5面目。とにかくボスが地獄です。そんなんありかーっというくらい攻撃してくださいます。耐えましょう。



登場するのは気持ち悪いキャラばかり

神殿のような雰囲気の6面。飛んでくる弾が圧倒的に多くなります。人間の情報処理能力を越えた画面が展開されますが，落ち着けば意外と避けられるものです。

このあとの7，8面は貴方の目でお確かめください。

まとめ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

遅ればせながらこのサイバーコア，PCエンジンからの移植なのです。X68000への移植にあたっては若干の手直しが入っているようです(背景の2重スクロール等)。またオリジナルにはなかった敵キャラが登場したりしています。しかしいちばんの相違点は，アイテムの出現率がかなり抑えられているということでしょうか？

オリジナル版では結構簡単にパワーアップできたのですが，X68000版では最高段階までパワーアップするのはなかなか難しくなっています。「爽快さ」ということにおいては，オリジナル版のほうがよかったと私は思います。シューティングはばりばり撃ちまくりたいもんね。

しかし手堅くまとまっていてかなり遊べるというのはさすがだと思います。ちょっとX68000用にはチープすぎるって気がしないでもないけどね。

虫の囁き

本文中であまりにも「虫，虫，ムシ……」といいすぎてしまったので，暗示にかかって気分が悪くなった方がいたらごめんなさい。まあ，何かの生態系にデザインの素をとるってのはシューティングにはよくあるパターンなんですよね。サイバーくんの場合あまりにもそれが顕著なので，つい口から出た（筆が走った）わけです，はい。だから虫嫌いの人も頭から「いやっ！」って決めつけないで，遊んでみてから決めてくださいね。

総合評価	0 5 10
操作性	★★★★★★★
BGM	★★★★★★
動き	★★★★★★★★
外骨格度	★★★★★★★★★★

スピードか，美しさか

Nishikawa Zenji
西川 善司

X68000ではまだ数の少ないフライトシミュレータ。発売されているものといえば，「遊撃王II」と「GUNSHIP」のわずか2つのみ。この「F15ストライクイーグルII」が発売されて3つになったけど，もっともっと出てほしいね。



嫌な予感がした。古いFLOAT2.XとOPMDRV.Xの組み込みメッセージが黒いCRT面に煌々と輝く。ローディング画面を見て不安がつのる。士官の手の動きを見て，冷や汗をかく。ゲームを始めると……。

「Wait a moment please」

後ろから殺気を感じ，振り向きざまに身構える私。

西川善司（以下善）「なんだ，お前は」

よく見ると，暗がりにターバンを巻いた，色黒のエキゾチックな男が立っていた。

「Hai！ My name is ペルシャ王子イイマース。My friends call me “プリペル”ト呼ブアルヨ」

善「どこの生まれだお前は。一体なにしに来たんだ，うちへ」

プリペル（以下プ）「アナタ，X68000ニ移植サレル外国ノゲーム，遅クテ，ツマラナイトキアル，思ッテマセーンカ？」

善「現に遅いものがあるだろうが」

プ「Oh! ソレ違イマース。アレハワザトデース。X68000ノアクションゲームミンナ動キ速クテ難シイネ。日本人働キスギネ。時間ニ追ワレテ生活スルノヤメテモラオト思ッテアアシタノコトヨ」

善「馬鹿いえ。ゲームはトロいのに，制限時間は現実時間で刻まれるもんだから，えらく時間に追われたゲームもあったぞ」

プ「ウーン，ナゼカ私ヲ心苦シクサセル言葉アルネ。But! Don't worry.『F15 STRIKE EAGLEII』ハoriginalモ動キ遅イシ，PC-9801版モアマリ速クナイカラ安心ネ」

善「なにが安心だか。じゃあ，このゲームの面白さを伝えてもらおうじゃないか」

プ「OK! マカセテクダサイ，ケツクサーイ。出掛ケルトキハ忘レーズニ」

ゲームスタートするぞ・ダダーン◆◆

プ「マズ，packageノ中ニハ88ページノ厚イmanualトkey refrence card，user登録ハガキ，ソレニナント私ノ故郷ノ中近東ノ地図ガ2枚モ入ッテマース。涙ナクシテミレナイシロモノデース」

善「地図ねえ」

プ「シカーモ」

善「クレアラシルか，お前は」

プ「X68000版ハCyber stick（Analog joy stick）ニ対応シテマース」

善「でも，要メモリ2Mバイトだって。別にオンメモリじゃないみたいだし，これってどういうことなんしょ？」

プ「Ha! ha! 素人ニハワカラナイ世界デース。flight simulation gamesハイロイロ大変ナンデース」

善「いろいろ大変ねえ。まあいいや。で，階級バッジが並んでるこの画面はなに？」

プ「one missionヲcompleteスルゴトニ得点ガ入リマース。ソノ得点ガアル基準ニ達スレバ昇級シマース。missionデsuperナ活躍スレバ勲章ガモラエマース」

善「RPGのレベルみたいなものか」

プ「撃墜サレテ死ンデシマウト経歴ガcome to an endデス。脱出シテ生存シテモ機体ヲ何回モ失エバ，desk work ニ飛バサレテ“THE END”デース」

善「いかにも外国のゲームらしいな。で，この怪しげなオッサンは一体誰なんだ？」

プ「gameノrankヲ選ブsceneネ」

善「このロッキーっちゅうのはなんだ？ シルベスター・スタローンか？」

プ「No! No! Rockyチガウネ。Rookie，新米ノコトヨ」

善「教科書どおりのつっこみありがとう。初めてのときはルーキーを選ぶのか」

プ「高levelデplayスレバスルホド得点ガ高イアルヨ。一方Rookieダト離陸モシナクテイイシ，初心者ニハオ勧メネ」

善「……にしてもこのオッサンの動きどうにかならなかったのかね。万博の日立館の受付ロボットのほうがいい動きしてたぞ」

プ「ナラバカワリニ私ノ動キ見テクダサーイ。ホラ，忍ビ足ノ動作ガsexyト評判デース。Disneyマッサオノanimation!!」

ペルシャ王子はいきなりお得意のくねくねアニメーションをご披露してくれた。

善「はいはい，立派立派。で，最後にミッション（任務）を選ぶわけね。システムソフトの『遊撃王II』みたいにストーリー性のあるシナリオなのかい？」

プ「ソレハ邪道ネ。『F15 STRIKE EAGLEII』ハsimulationデース。ダカーラ同ジmissionヲ繰リ返シ練習デキルワケデース」

善「わたしゃ，『遊撃王II』みたいのが好きだけどね」

プ「missionハLIBYA，PERSIAN GULF，VIETNAM，MIDDLE EASTノ4ツガアリマース。私ノ故郷ノ近クガアッテウレシーデース。デモ私，戦争反対，豚タベナーイ」

善「たった4つのミッションしかないの」

プ「Oh! シカシ4種類ノ難易度ガアルシ，攻撃目標モヤルタビ変ワルカラ数エキレナ



X68000用 5"2HD版2枚組 10,800円（税別）
マイクロプローズジャパン ☎0423(33)7781

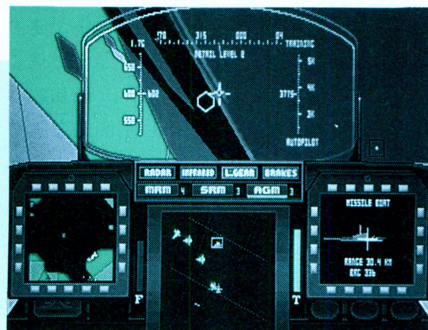


コキコキ動くオヤジが目標を指示する

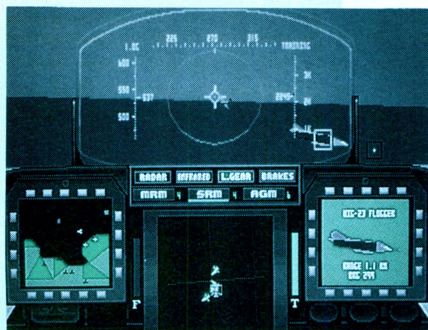
イホドノmissionガアルコトニナルヨ」
善「なんかずるい数え方だな。KAWAIのリズムマシンXD-5みたいだ」
プ「細カイコト気ニシナイ。ツマランコトイッテルト，モウ石油輸出シナイアルヨ」

飛ぶぞ・ビュビューン◆◆◆◆◆◆◆◆

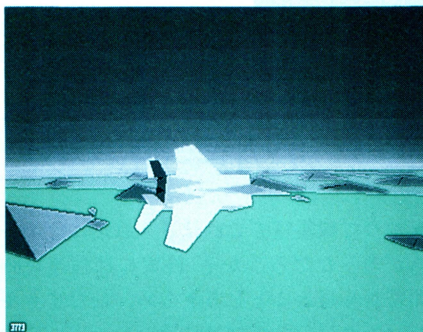
善「で，ミッションを選択したら飛ぶことになるわけね」
プ「Yes! 今，Rookieヲ選択シテマスカラ離陸ノ必要モアリマセーン」
善「やっぱりちょっとのろいなぁ。動きも粗いし」
プ「Oh! dear. 島国ノ人々，細カイコトバカリ気ニスルネ。Too bad. 大陸ノ人ミタク大キナ気持チ，持ツヨロシ」
善「でも，なんとかならない？ ビデオのコマ送りみたいで気持ち悪いよ」
プ「安心シテクダサーイ。[CTRL]＋[D]ヲ押スト画面ノdetailヲchangeスルコトガデキテ処理速度ヲ上ゲラレマース」
善「色数を落とし，表示する物体数を減らしたり，形状を簡単にして処理を軽くするのか。考えたね」
プ「ホーラ，速クナリマシタ」
善「でも，DETAIL0は本当に0って感じ」
プ「デモ操作ハ本格的，実際ノF15ニ乗ッタ気分ヨ。youモace striker，TOP GUNヨ」
善「本格的って，操作が複雑なんじゃないだろうな？」
プ「ダイジョーブネ。イチバン絵ノキレイナDETAIL4ニシテオケバ，key referance



DETAIL2の画面はこんな感じ



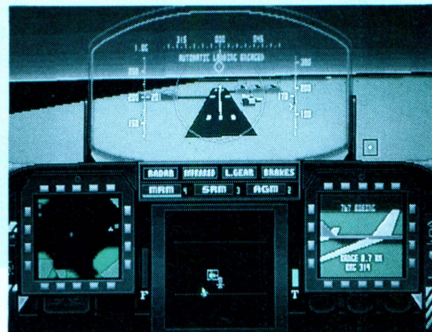
DETAIL0だと動きは速いが何もない


DETAIL4で外部視界にしてみる

cardヲ見ナガラplayシテモ全然平気ネ」
善「計算された遅さだといいたいワケね」
プ「Of course，モチコース。『モチコース』ハ『モチロン』ト『Of course』ノ合成語ネ。念ノタメ」

攻撃するぞ・バババーン◆◆◆◆◆◆◆

善「このレーダー上のピカピカ光ってるオレンジの十字は？」
プ「コレ？ 攻撃目標ノ位置示シテルアルネ。攻撃目標ハ2ツアッテ，両方破壊シタラ基地ニ帰還シテイイアルヨ」
善「ああん。慣れてないから操作が複雑で目標まで行けないよ」
プ「youのようなstupid boyノタメニAUTO PILOTモアリマース」
善「ほほー」
プ「目的選択keyデ飛行目的ヲ選ンデ，AUTO PILOTヲonニスレバOKデース」
善「こいつは便利だね。あとはテレビを消してお茶でも飲んでればいいのかい」
プ「ソウイウワケニモイキマセーン。youニトッテハ攻撃目標デモ敵ニトッテハ重要ナ軍事拠点デース。デスカラyouノ進入ヲ阻止スベク敵機ガwaiting for you」
善「お前本当に中近東の人間か。話し方がルー大柴みたいだぞ。で，いちいちオートパイロットを解除したりしなきゃいけないの？ めんどくせーな」
プ「移動keyニ触レバ自動的ニ解除サレマース。攻撃ハ敵ヤ敵ノ位置ニアワセテ武器



着陸もオートにできるので楽チン

ヲ選択シgun sightニ敵影ヲsetシテtriggerヲ引クダケデOk。Simpleデショウ?」
善「ふーん。思ったほど操作は複雑じゃないんだね」

帰還するぞ・バビューン◆◆◆◆◆◆

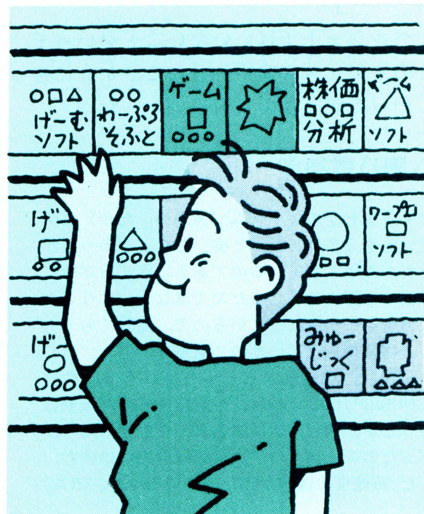
任務を終えて帰路につく。
善「フライトシミュレータってさ。着陸が難しいんだよね」
プ「ソノ心配イリマセーン，カッパエビセーン。先ホド攻撃目標ヲsetシタ要領デ味方ノ基地ニ進路ヲsetスレバ，アトハAUTO PILOT ONデ自動的ニ帰還シマース。ダケド依然ト敵ノ追撃ハ続キマスカラ油断ハ禁物，男ハきん持ツ」
善「……。『きん』で思い出した。俺トイレ行ってくるわ。ええーと，ESCでゲームを一時停止っと」
プ「Oh! no! ソレ一時停止ト違イマース。脱出KEYデース。PAUSEハ[CTRL]＋[P]デース。Oh! my god」
善「早くそれをいえよ。普通のゲームじゃESCは一時停止だろ」
プ「Ahh! アナタ音速ノ状態デ脱出シタネ。pilot，脱出ノトキニ即死シタアルヨ。死亡シタカラ登録抹消。モウ一度名前登録カラヤリ直シ。アナタ愚カ者ネ」
善「……」
ペルシャ王子はいいたいことをいって，とあるディスクの中へ「にこやか」に帰っていったが，私の表情は硬かった。

速さよりもシステム周り重視型

たしかにこのゲームのスピードは速くない，というよりキー入力に対する反応が鈍いために快適な飛行感覚という感じではない。しかし，もともとマイクロプローズは，そのへんよりもゲームシステムの充実度で売っているので，戦術的な楽しみを求める人にはいいだろう。
また，マイクロプローズのひとつの特徴であったキーボードオーバーレイがなくなったことは評価に値する。別に経費節減というわけではない。わかりやすい操作体系にされたのである。フライトシミュレータの操作体系は“わかりやすく，なおかつ他ゲーム（海外の，だが）との統一を考えてほしい”と書いたことがあったが，それが実現されているのである。別に日本版に移植する際にそうなったのではなく，オリジナルの時点でそうなっているので，海外でもそのことは指摘されていたのであろう。
しかし，やはりスピードは気になる。前作「GUNSHIP」はミッションや演出の部分で不満はあったが，スピードはまあまあで動きも滑らかだったので，ヘリコプターの操縦感覚は味わえた。解像度の違いが出たのであろうか。(R.A.)

AFTER REVIEW

ひとりでやっているとそれほど面白いこともないんだけれど，何人か集まって対戦すると至上の喜び，というゲームがある。そのうちのひとつ，しかも最高ランクに位置するのが，この「ボンバーマン」だ。



ボンバーマン

▶かつて編集室でこれほど入れ込んだゲームがあっただろうか？ 終電には程遠い時間帯から帰ることをあきらめ，ひたすらボンバーマンのためだけにマシンルームに泊り込む。合言葉は「徹ボンしない？」。

ブラックコーヒー片手にワープロとにらめっこをしているほかのライターを尻目に「たま～っ！」「ひぃ～！」と奇声を発するOh!Xのライターたち。ああ，極上のとき。ボンバニアンたちの長い夜は続く。

(赤いボンバーキングこと，S.K.)

▶ボンバーマンの魅力はなんといっても「対戦」でしょう。だって友達が4人集まって同時プレイできるゲームなんて，ほかにはないじゃありませんか。などといいつつも，実は私，編集室では「青いバカ」などと罵られ，戦えども戦えども連戦連敗なのです。「ボンバーマンやろうよ」というと，仲間はまるで，カモがネギを背負って，おまけに鍋まで担いでやってきたような顔をされてしまう(とほほ)。それでも大人数でできるゲームって楽しいよね。パソコンを大勢で囲んでわいわいやるゲームっていうのは，それだけで楽しいもんだ。また，自分のプレイするボンバーマンをいつも決めておくのもいいね。ちなみに「青いバカ」の私は当然，青いボンバーマンでプレイしているのだ。 (毛)

▶こんなに面白いゲームがX68000のユーザー総数の4分の1も売れていないなんて不思議だ。 (善)

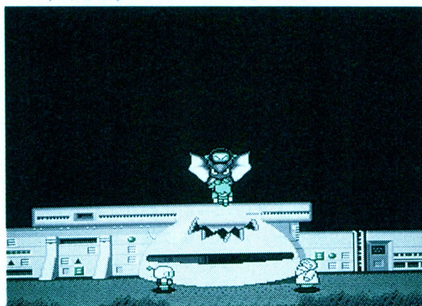
▶敵は人間だ！ これがボンバーマンの神髄であろう。対戦プレイ，もとい，対戦バトルロイヤルである。騙し，陥れ，そして自分だけが生き残る快感は「私が女王さまよ。ほ～っほっほっほっ」という世界に通じる危ない快楽といっても過言ではない。ただ，元のPCエンジン版を尊重したせいか，4人同時までしかサポートしていないために，必ず中央から「お化け」が出てくる。やっぱりここは「コンピューターボンバーマン」を出してほしかった。どうせならX68000版ならではの$+\alpha$がほしかったところである。ゲームセンター版のようにひとりでも対戦モードで遊べたらと思うと，ちょっと残念だ。 (八)

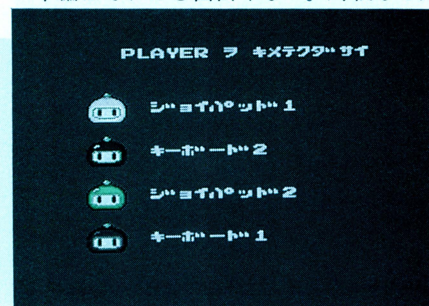
▶寮に住んでいるような人の場合（僕がそうだが），夜中に暇を持て余した連中が寄り集まると，数時間の不毛なひとときを過ごしてしまう。対戦ボンバーマン。最初のうちは編集部での経験を生かし圧倒的な強さで初心者をいたぶっていた僕も，そのうち「恐るるに足らぬ」と呼ばれるようになってしまった。あの手この手で相手をパニックに陥れる。ドクロマークを食ってゲームをひっかき回すのも流行った。キメ技がぴたりと決まると気持ちいい。キメられるとけっこう悔しい。

本編はそれほど面白くない。対戦も4人









でないとつまらない。しかし４人だと，これをやらずに死ねようかというくらい面白い。ゲームデザイン上のほんのちょっとした練り方と，プログラミング技術が多少足りない気もするが，それを差し引いても今年いちばん熱中したゲームのひとつだ。

(A.T.)

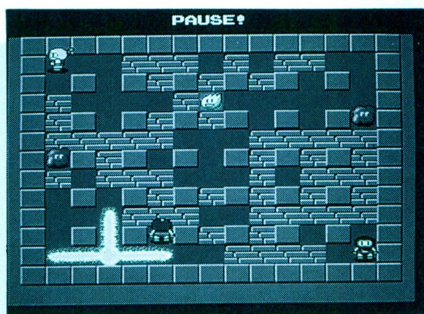
▶自分の仕掛けた爆弾にひっかかって死ぬ。こんなにくやしいものはない。というのが，ボンバーマン（１人プレイしかできないファミコン版の場合）の魅力だった。そのボンバーマンに対戦プレイがついて，自分の爆弾以外にも気を使わなくてはならなくなったので，緊張感が２倍，いや10倍近くにまで高まっている(当社比)。始めのうちは爆弾の威力が弱いから，あまり爆弾の火を避けることを気にせずプレイしても大丈夫。しかし，みんながみんなアイテムを取って爆破範囲が広くなると，もうたいへん。自分の仕掛けた爆弾，相手の仕掛けた爆弾が入り乱れ，画面は「ボカボカ，ボカボカ」と爆弾の火が舞い踊っている。人数が減って，爆破の影響で画面もすっきりしてしまうと，そこからはブロックの陰に隠れ，相手の動きを読みながら，なんとかハメようと爆弾を工夫して置いていく。うーん，派手な要素と陰険な要素が入り交じる対戦プレイ。やっぱり燃えるね。こんなに簡単なシステムでこんなにハマれるんだよ，という好例だな。　泉　健二(19)埼玉県

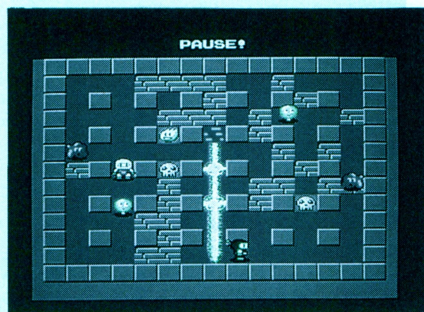
▶８月半ば４人の戦士が集まった。もちろん夏休みの課題を賭けたボンバーマン大会のためである。ルールは２泊３日耐久無制限デスマッチ。ぶっとおしで対戦をやり続けたが126試合目でダウン。勝率による課題分配をするときには，みんな頭がボロボロになっていた。結局，いちばんにはなれなかったがまあいいや。う～ん，若さっていいなあ。　橋本　和幸(15)東京都

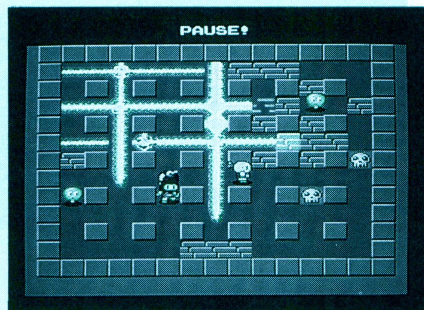
▶皆が皆「ボンバーマンは対戦が面白い」といっているけど，１人プレイでも十分面白いじゃないですか。PCエンジン版からの移植とあって，パスワード機能もあるし，長く遊べると思うんだけどなあ。僕は気に入っていますよ。いや，決して「１人プレイがおもしろい」といっているのは，一緒に対戦してくれる友達がいないからじゃないぞ。本当ですからね。

中山　肇(18)石川県

▶対戦ではプレイヤーの個性がモロに出るのがいいですね。他人を相手にせずひたすらアイテムを集めるヤツ，他人をはめようとして自爆する馬鹿，やたらに病気を取りまくって人にうつそうとするヤツ。それぞれ，思い思いに美しい爆弾の華を咲かせるのは楽しいったらありゃしない。そして，このゲームを徹夜でやったあとには，完全にみんなの性格が変わってしまった。う～ん，暇を持て余していたとき，ちょっと手をつけたボンバーマンにこんなにはまるとは，お釈迦様でもわかんなかっただろうな。　佐藤　崇(20)神奈川県









発売中のソフト

★ダーウィンズジレンマ	スタークラフト	
X68000用		5″2HD版 9,800円(税別)
★麻雀マスター	ブラザー工業(TAKERU)	
X68000用		5″2HD版 7,800円(税込)
★パワーモンガー	イマジニア	
X68000用		5″2HD版 12,800円(税別)
★F15ストライクイーグルII	マイクロプローズジャパン	
X68000用		5″2HD版3枚組 10,800円(税別)
★ラストバタリオン	スティング	
X68000用		5″2HD版 8,800円(税別)
★ブリッツクリーク	システムソフト	
X68000用		5″2HD版2枚組 9,800円(税別)
★フェアリーランドストーリー	SPS	
X68000用		5″2HD版2枚組 6,800円(税別)
★NIKO²	ウルフ・チーム	
X68000用		5″2HD版 7,800円(税別)
★ディノランド	ブラザー工業(TAKERU)	
X68000用		5″2HD版 7,800円(税込)
★ノーブルマインド	ブラザー工業(TAKERU)	
X68000用		5″2HD版3枚組 5,900円(税込)
★サイバーコア	SPS	
X68000用		5″2HD版 7,800円(税別)

新作情報

★ヴェルスナーグ戦乱	ファミリーソフト	
X68000用		5″2HD版 価格未定
★プロサッカー68	イマジニア	
X68000用		5″2HD版 9,800円(税別)
★アルシャーク	ライトスタッフ	
X68000用		5″2HD版 9,800円(税別)
★スターウォーズ	ビクター音楽産業	
X68000用		5″2HD版 7,200円(税別)
★ノア	M.N.Mソフトウェア	
X68000用		5″2HD版 7,200円(税別)
★飛翔鮫	金子製作所	
X68000用		5″2HD版 8,800円(税別)
★大戦略III'90	システムソフト	
X68000用		5″2HD版2枚組 9,800円(税別)
★スーパー上海ドラゴンズアイ	ブラザー工業(TAKERU)	
X68000用		5″2HD版 7,800円(税込)
★SPINDIZZY II	アルシスソフトウェア	
X68000用		5″2HD版 価格未定
★PITAPAT	ビクター音楽産業	
X68000用		5″2HD版2枚組 6,800円(税別)
★ゼノン2	エピック・ソニー	
X68000用		5″2HD版 価格未定
★出たな!! ツインビー	コナミ	
X68000用		5″2HD版 9,800円(税別)
★ジェノサイドII	ズーム	
X68000用		5″2HD版 8,800円(税別)
★シムアース	イマジニア	
X68000用		5″2HD版 12,800円(税別)
★レミングス	イマジニア	
X68000用		5″2HD版 9,800円(税別)
★F29 RETALIATOR	イマジニア	
X68000用		5″2HD版 価格未定

Oh!X 4周年記念

愛読者
特別モニタ大募集

月日が過ぎるのは早いもので，本誌Oh!Xは今年４周年を迎えることとなりました。誌名変更してからもう４年だなんて，なんだかほんとにあっという間で信じられない気持ちと，愛読者の皆様への感謝の気持ちでいっぱいです。スタッフ一同お礼を申し上げます。皆様の励ましやお叱り，貴重な意見などに支えられて，私たちはこれからもますますよい雑誌に，と頑張っていくつもりです。ということで，今後ともよろしくね。

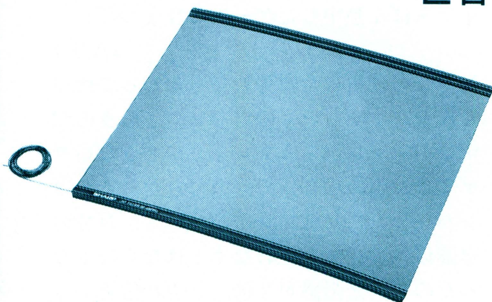
さて，今年もシャープさんからたくさんの商品をご提供いただきました。シャープさん，いつもいつもありがとうございます。

でもって，今回は，タイトルにもあるとおり，すべてモニタ募集ですので，当選なされた方は，場合によってはモニタレポートを提出していただくことがありますので，あらかじめご了承のうえ，ご応募お願いします。

それでは皆さん，応募方法をよく読んで，レッツらゴー！

1 CRTフィルター (BF-68PRO)

19,800円(税別)　2名



可視光線の約60%を吸収して，目の疲れをやわらげてくれます。長時間ディスプレイを眺める人に。

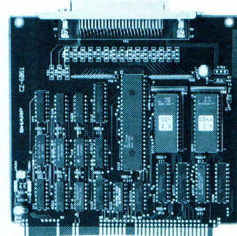
2 マウス・トラックボール (CZ-8NM3)



9,800円(税別)　2名

マウスとトラックボール，両方の機能を兼ね備えたスグレもの。ゲームやCGなどにも最適です。

3 SCSIボード (CZ-6BS1)

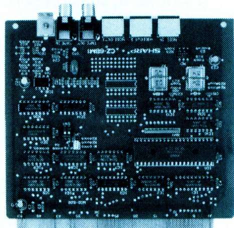


SCSIインタフェイス周辺機器を接続するためのもの。SCSIユーティリティソフトがついています。

29,800円(税別)　1名

4 MIDIボード (CZ-6BM1A)

26,800円(税別)　1名

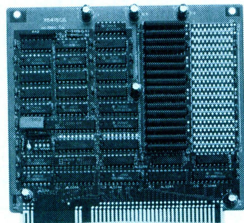


X68000とMIDI楽器の間で情報をやりとりするためのボード。コンピュータミュージックをやる方に。

5 2Mバイト増設RAMボード (CZ-6BE2)

2Mバイト拡張できるRAMボードです。使用に際して1Mバイト増設RAMボードが必要となります。

79,800円(税別)　2名



6 NEW Print Shop PRO-68K ver.2.0

X68000用　5″2HD版

20,000円(税別)　1名



ver.1.0を高速化。オリジナルカードなどが簡単に作成できるポップアートツール。カレンダー作成機能もある。

7 C compiler PRO-68K ver.2.0



X68000用　5″2HD版

44,800円(税別)

1名

ソースコードデバッガのほか，MAKE，ライブラリアンなどの開発機能を付属したCコンパイラソフト。

応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ、希望する番号をアンケートはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1991年12月18日の到着分までとします。当選者の発表は1992年2月号で行います。

8 Multiword

X68000用
5″2HD版

32,000円(税別)



2名

マルチウィンドウ編集とテキスト編集の2つのインタフェイスをサポートした多機能ワープロソフト。

9 ダッシュ野郎

X68000用
5″2HD版2枚組

8,800円(税別)



3名

数々の障害物をよけつつゴールを目指すバイクラリーゲーム。サイバースティック対応。

10 中華大仙



X68000用　5″2HD版　7,900円(税別)

3名

アーケードから移植のシューティングゲーム。「西遊記」を思わせるキャラクターが特徴。

10月号プレゼント当選者

1黄金の羅針盤（埼玉県）森雅秀（大阪府）福山季男（福岡県）堀幸司 **2**ファランクス（岐阜県）遠藤正彦（滋賀県）鶴田雄三（福岡県）梶谷太郎 **3**スターモビール（埼玉県）小川毅（愛知県）笹田泰治（福井県）岡部誠 **4**ALL THAT RPG（東京都）岩瀬達彦　橋本善成（神奈川県）林広国（広島県）谷川正洋（愛媛県）住友智代ほか5名 **5**清涼飲料水（群馬県）久保田智久　（敬称略）

以上の方々が当選されました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，公正取引委員会の定めにより，今回のモニタ募集に当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

謎のV70ボードを追う

V70とは何者か?

Nakano Shuichi 中野 修一

X68000の前に突然現れた謎のCPUボード,「V70＋AFPPアクセラレータ」。現段階ではまだまだ謎に満ちた部分を秘めており正体は判明しない。ここではとりあえず,その「頭脳」である国産32ビットCPU,V70を紹介してみたい。

11月号でのアクセスの広告に驚かれた方も多いと思います。「あなたのX68000がSuperワークステーションに」という触れ込みでV70アクセラレータの発売がアナウンスされていたからです。

これはまったく「寝耳に水」といえます。ここ3年間のマイコンショウの取材記事で,なぜか鳴かず飛ばずのVシリーズの写真が掲載されていたことはまったくの偶然と思っておいてください。

現時点ではまだ広報資料さえない状況なので詳細については来月または再来月にならないとお伝えできません。広告からだけでもあらかた推測はできるのですが,今月はとりあえず,V70というCPUについて紹介しておきましょう。

日電のVシリーズ

セガは「ラッドモビール」からシステム32という32ビットシステムボード上でゲーム開発を始めた……というニュースが伝わったのは91年の春のことでした。新しい「システム32」はそれまでの68000を使ったシステムボードの5倍以上のパワーを持つとされていましたね(CPU性能だけとは思えないが)。

そこで使われていたCPUはNECのV60(16MHz)というものでした。V60は内部32ビット,外部16ビットバス仕様のCPUですので,68000と同じ構成なのですが,メーカーは32ビットCPUと呼んでいるようです。確かに,V60/70/80シリーズでは,

8ビット ＝ 1バイト
16ビット ＝ 1ハーフワード
32ビット ＝ 1ワード

というふうに呼ぶようですけど。

さて,NECのVシリーズとはいっても,V20/30/50のような16ビット仕様のものとV60/70/80の32ビット仕様のものではまったく様相が違います。一部にそのあたりを勘違いしている読者の方もいるようなので少し整理しておきましょう。

*　　*　　*

たとえば,V70を68000ファミリーと比較してみましょう。どちらも32ビットアーキテクチャを基本に設計され,豊富な汎用レジスタと充実したアドレッシングモードを備えています。まず16ビットバス版の68000やV60が作られ,完全32ビット版の68020/30,V70,そして高速版の68040,V80が作られていく経緯もなぜか似ていますね。

68000/10/20/30/40のレジスタ構成はどれもほぼ同じで,ご存じのようにデータレジスタが8本,アドレスレジスタが8本となっています。

V60/70/80では汎用レジスタが32本です。用途に制限はありません。

アドレッシングモードは68000が10種類,68020以降が18種類。Vシリーズでは21種類です。「PC相対2重インデックス」とか,レジスタを使わず間接アドレッシングのできる「直接アドレス間接インデックスつき」などが目を引きます。ちなみに,ディスプレイスメントには8/16/32ビットが扱えます(68Kシリーズも68020以降は32ビットまで可能)。

ほぼ完全に直交した2オペランド対称の命令形式はソース,デスティネーションのいずれも独立したアドレッシングモードが自由に選択できます。

*　　*　　*

ミニコンや68000シリーズを意識してはいるのでしょうが,驚くほど綺麗な構成を持ったCPUです。「簡単にハンドアセンブルできる」というのもあながち噂だけではないのでしょう。

単に綺麗にまとまっているだけでなく,高機能命令も備えています。メモリ上の任意の位置にある,1ビットから4Gビットまでのビット列の転送や論理演算が1命令でできるとか,ストップ文字を指定して文字列を比較する専用命令があるとかいった具合です。

こうなると68000のプログラムを移植することは非常に簡単に思えます。最大の違いは「エンディアンの違い」(ビット/バイトの並び方向)ですが,どうやらちゃんとビット/バイト順を入れ替える専用命令も用意されているようです。

その他,いわゆる高級言語指向の命令もかなり整備されています。しかし最近はRISCのようにコンパイラ技術とハードウェアが並行して開発されるのが流行ですから,「コンパイラが作りやすいこと」と「優秀なコンパイラがあること」にはあまり関係がないようです。むしろアセンブラを使うユーザーに喜ばれそうです。

ハードウェアの特徴

ハードウェア的な特徴を見てみましょう。MMUを内蔵していますので本格的なOSを稼働できます。MMUが扱う領域は3階層に分けられアクセス空間全体が4セクションに1セクションが1024エリア(1Mバイト)に1エリアが256ページになっています。メモリプロテクションなどはエリア単位に行われます。ページというのはメモリスワッピングの単位と考えていいでしょう。

最新のチップに比べるとキャッシュサポートの面で見劣りがしますが,6段パイプライン処理など68000の2ワードプリフェッチ式のパイプラインに比べ,命令の並列実行度が高くなっています。広告にあった「最速10MIPS」というのはあまりあてにならない数値ですが,実測値でレジスタ間の演算などの基本的な部分は2クロックで処理されているというだけでも,クロック計算しながら68000のアセンブラを使っていた人には魅力的に映ることでしょう。20MHzというクロックともあわせて68000よりはるかに高速なことは確かです。チップの基本性能では68030以上でしょう。

シフト演算についてはバレルシフタの採

用で実行時間はシフトするビット数には関係なくなっています。

上位のV80となると，大幅なワイヤー化により基本命令はすべて2クロック均一で処理し，分岐予測機構を備え，実にV70の倍の性能を発揮します。1命令の機能を考えると2クロックという実行ステップがいかに驚異的かがわかります（キャッシュがヒットすればレジスタもインデックスつきもアドレッシングモードによらず同速度)。ただ，V80は電気食いで発熱も凄いとは聞きますが……。とりあえずV70でも，それ以上の性能のCISC型といえば「総当たりする魔法のキャッシュ68040」と「力技1クロック実行80486」と一部のTRONチップくらいのものです。

V70自体も浮動小数点演算命令を備えていますが，今回のアクセラレータにはAFPP (ADVANCED FLOATING POINT PROCESSOR) も備えられています。現在X68000上で動作する数値演算プロセッサ68881も実際はかなり高性能な石です。しかし，X68000とのインタフェイス部分でかなりのオーバーヘッドがかかるため現状ではその能力をまったく発揮できていません。ちゃんとコプロセッサとして動作できれば，それだけで格段に性能は上がります。もちろんAFPPはV70のコプロセッサとして接続されています。石自体の基本性能がさらに高いので大量の実数演算を高速に処理できます。

ほめるところが多くなってしまいましたが，68000よりは5，6年新しいアーキテクチャで設計されているのだから，ある意味では当然なのかもしれません。しかし，それを加味してもよく作ってあるCPUであることには変わりないでしょう。

開発された時期がCISCの全盛期だったためか，RISC全盛の現在ではパッとしませんが，構成的にはもっとも完成されたCISC型CPUと呼べる内容です。V60/70/80シリーズはユーザーがアセンブラを使う気になる最後のCPUなのかもしれません。

V70アクセラレータとは?

一般的なアクセラレータでは，現在使用しているCPUの代わりにより高速なものを使うのですが，それにはオブジェクトレベルの互換性が必須となります。V70ボードの場合，既存のX68000用ソフトウェアが速くなることはないはずです。

将来的にはわかりませんが，当面は68000との並列動作が主体となるものと思われます。

しかし，現在V70用で実行できるソフトがまるでないわけではなく，Vシリーズには V30のエミュレーションモードもついていますので8086用の基本的なプログラムは実行できるはずです(理屈のうえでは)。がんばれば以前アクセスが販売していたCONCERTOのようにMS-DOSエミュレータを稼働させることも不可能ではないでしょうし，8080エミュレーションが使えれば超高速のCP/Mマシンにもなるでしょう。もちろん，この場合V70のスペックはほとんどが眠ってしまいますが……。

最低限の開発環境は揃っているようなので，68000のアセンブラを使える人ならソフトの開発は容易でしょう。基本ソフトウェアの開発はハドソンが担当しているようですので，C言語のプログラムはほとんど問題なく実行できると思われます（ライブラリの詳細は不明)。

X68000の場合，

BASIC→C言語→実行形式

というかたちが確立されていますから，ユーザープログラムのほとんどは比較的簡単にアクセラレータの恩恵にあずかることができると予想されます（まだまだ予断を許しませんが)。また，「32ビットの汎用レジスタがたくさん」というのはGCCのデフォルトオプティマイズがもっとも得意とするタイプでもあります。いろいろと楽しみは多そうです。

価格，動作形態，ソフトウエアの詳細など気になる情報はまだ入っていません。入りしだいレポートしたいと思います。

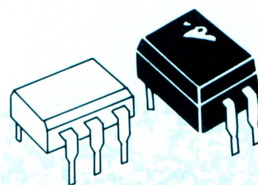
図1　レジスタセット

プログラムレジスタセット (31 … 0)

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29 (AP : Argument Pointer)
R30 (FP : Frame Pointer)
R31 (SP : Stack Pointer)
PC (Program Counter)
PSW (Program Status Word)

特権レジスタセット (31 … 0)

LSP (Interrupt Stack Pointer)
L0SP (Level0 Stack Pointer)
L1SP (Level1 Stack Pointer)
L2SP (Level2 Stack Pointer)
L3SP (Level3 Stack Pointer)
SBR (System Base Register)
TR (Task Register)
SYCW (System Control Word)
TKCW (Task Control Word)
PIR (Processor ID Register)
PSW2 (Program Status Word2)
ATBR0 (Area Table Base Register0)
ATLR0 (Area Table Length Register0)
ATBR1 (Area Table Base Register1)
ATLR1 (Area Table Length Register1)
ATBR2 (Area Table Base Register2)
ATLR2 (Area Table Length Register2)
ATBR3 (Area Table Base Register3)
ATLR3 (Area Table Length Register3)
ADTMR0 (Address Trap Mask Register0)
ADTMR1 (Address Trap Mask Register1)
ADTR0 (Address Trap Register0)
ADTR1 (Address Trap Register1)
ADTMOD (Address Trap Mode Register)



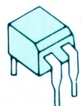
ハードウェア工作入門《18》

ハイテクタンク製作（発展編）

Misawa Kazuhiko
三沢 和彦

今月号ではパトリオットのマル秘システムである，自動追尾システムを設計していきます。なにやら難しそうな感じがしますが，回路自体は光センサー2個を使った簡単なもの。ノイズ対策などの解説を読みながら，さくさく理解できるでしょう。

いよいよパトリオット製作も詰めの段階に入ってきました。先月までに基本的な動力部分は完成させましたが，今月は予告どおり，パトリオットのマル秘システムの種明かしをしましょう。これまでにないほどの簡単な回路を追加するだけで，パトリオットが人工知能（？）を持ったスーパーハイテクマシンに生まれ変わります。



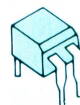
マル秘システムとは？

最初に，「パトリオット」命名の秘密からお教えします。そもそも，「パトリオット」というのは，湾岸戦争のときに一躍有名になった対空自動追尾迎撃ミサイルの名前です。このミサイルは敵から発射された攻撃用ミサイルに対し，その飛んでくる方向及び速度などを自動追尾して敵ミサイルを空中で撃ち落とすものです。自分の陣地に攻撃ミサイルが達する前に破壊してしまうので，迎撃に成功すれば受ける被害が最小ですむのです。

そして，その最大の特徴は自動追尾という点であり，飛んでくる攻撃ミサイル1発ごとにパトリオット1発を撃って防御していきます。この自動追尾には，敵の位置を検知するセンサーシステムと検知した敵の位置に，自分が向かうようにコントロールする駆動システムとの組み合わせが用いられているのが一般的です。これはちょうど，この連載でこれまでに解説してきたセンサー回路とモーター制御とを組み合わせるだけで基本的な自動追尾システムが実現できてしまうのです。

そこで，今回製作したハイテクタンクの「パトリオット」にも自動追尾システムを搭載することにしました。これが，パトリオット命名の秘密です。といっても，空を飛んでくる物体を自動追尾するには，それこそ最新兵器テクノロジーの粋を集めないと構成できません。そこで今回は，タンクに向けられたサーチライト（辺りが暗ければ懐中電灯で十分）の光の方向を自動探知し，その方向に自動的に進んでいくように設計することにします。

光源の自動探知には，人間のように目（光センサー）を2つ持っていればOKです。光センサーそのものについては，この連載の「センサー回路」（1991年1～4月号）で詳しく取り扱っているので，基本的な概念は皆さんの頭に入っていることと思います。ところで今回扱う光センサーはそのときに解説したフォトダイオードとは違って，CdS光電セルと呼ばれるものを使用しますが，詳しい違いは後で述べることにします。ここでは，光センサーを2個使った光源自動探知システムの基本的な仕組みを押さえておきます。



自動追尾の仕組み

光源自動探知の仕組みは図1を見てください。パトリオットには，横に2つ同等の光センサーを並べて取り付けてあります。光源に対して，まっすぐ向いているときには左右の光センサーに入ってくる光の量は原理的には等しくなっています。

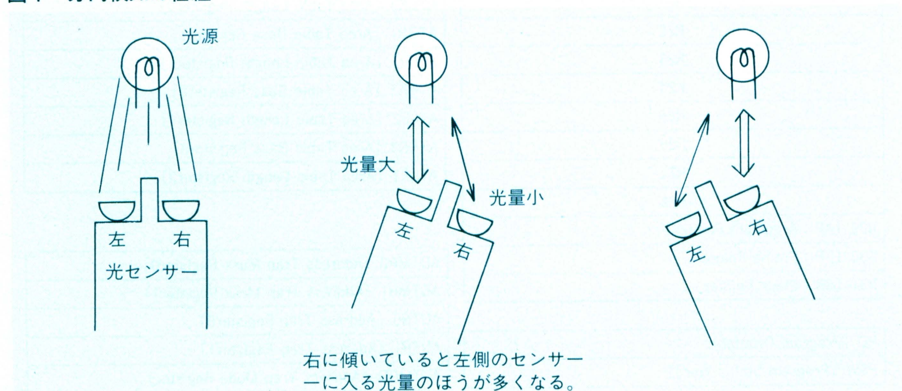
また，どちらか横のほうに逸れて向いているときには，光源に近い側の光センサーに入ってくる光の量が多くなるはずです。ですから，2個の光センサーに入ってくる光の量を常に監視して比べていれば，左右に入ってくる光量の大小関係でパトリオット本体の光源に対する向きがわかるのです。

実際問題として，両方のセンサーに入ってくる光量がまったく等しいというのはかなり限定された場合で，パトリオットが動いている限り，左右どちらかの光量が多い場合が一般的です。したがって，向きのモニターも光源に対して右に向いているか左に向いているかの二者択一で判断すればいいことになります。

このシステムを使って光源の方向を自動追尾していくには，自分が光源に対してどの方向を向いているかを前述したように検知した後に，モーターの駆動をコントロールしてパトリオット本体の向きを変えればよいわけです。パトリオットは左右旋回できるように設計されていますから，向きを変えるのは簡単です。右（左）に向きすぎていると判断したら，左（右）に旋回すれば正しい方向に向きを変えることができます。

コンピュータとの関係は，I/Oインタフェイスを理解している皆さんには非常にやさしい問題だと思います。光センサー回路で光源の方向を検知すると，そのデータは入力インタフェイスを通じてコンピュータに入ってきます。コンピュータはそのデータから光源に対する自分の向きを判断し，

図1　方向検知の仕組

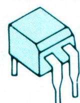


右に傾いていると左側のセンサーーに入る光量のほうが多くなる。

光源の方向に向きを変えるための動作を決定します。

この条件判断の部分をCPUが行っているのです。動作が決定されたら，出力インタフェイスを通じて実際にモーターを駆動し，自分の向きを変えます。この処理を繰り返すことによって，常に光源を自動追尾することができるのです。

この，センサー→入力インタフェイス→CPU→出力インタフェイス→機械制御という処理の流れは現在では身近にもよく見られる自動制御の鉄則ともいえましょう。たとえば，エアコンによる室内の温度調整にしても，温度検出→インタフェイス→CPU→インタフェイス→冷房機（暖房機）のON/OFFという一連の流れに沿っています。このハードウェア工作入門でも自動制御のほんの基礎の部分を実際に試してみることができたわけです。



センサー回路の実際

では，実際にパトリオット本体に搭載するセンサー回路部分を検討していきましょう。図2がその回路図ですが，始めに今回の光センサーであるCdS光電セルについて説明しておきましょう。

CdS光電セルは光が当たると光強度に応じて電気伝導度が変化する素子です。構造は図3のように，極性のない2本の端子の出た基盤上にCdS(硫化カドミウム)という半導体粉末を焼結させたものです。

原理は光導電（photo-conductive）効果といって，CdSに光が当たると光エネルギーによって伝導電子が生成され，その伝導電子が電気伝導に働くという仕組みになっています。図4はCdS光電セルに対する照度と抵抗値の関係を示したものです。光が当たると伝導電子が生成されるため，光量が多いほど伝導度がよくなり，すなわち抵抗が減少します。そこでセンサーの使い方としては，光量に応じた可変抵抗としてその抵抗値（電気伝導度）をなんらかの方法で検出することになります。

今回は回路図中に示されているように，LM393というICが検出器になっています。このLM393というICは，コンパレータと呼ばれるオペアンプの一種です。実際には1個のLM393にコンパレータ回路が2個入っています。回路図記号がオペアンプとそっくりなのに気づくでしょう。

これには，2つの入力端子＋と－があり，－端子に基準電圧をかけておきます。動作は，＋端子の入力電圧が基準電圧より大きいときに出力がH，小さいときにはLという実に単純明快なものです。原理は＋端子と－端子の電圧差を増幅し，わずかな電圧差でもその大小関係を出力することができるというもので，基本的にはオペアンプと変わりありません。オペアンプについても以前詳しく解説したので，本紙バックナンバーを参考にしてください。

今回の自動探知システムでは向きの判断は左右のどちらが大きいかの判断だけですから，コンパレータの出力のH・Lに右・左を対応させることにします。

次に，基準電圧とセンサーからの出力電圧との作り方ですが，これはどちらも可変抵抗の組み合わせでできています。基本的には図5の回路を設計します。これは，この連載のA/Dコンバータの応用で製作した簡易ジョイスティックとまったく同じ分圧器というものです。

すなわち，分圧器とは2つの抵抗R_1，R_2を図5のように接続し，両端に定電圧（電源電圧）をかけておき，2つの抵抗の中点から出力端子を出してその電圧を求めると，R_1とR_2の比で分割された電圧値が出力されてくるというものです。そこで，可変抵抗を使うことによって抵抗値が変わると当然2つの抵抗値の比も変わるため，分圧器

図2　センサーシステムの回路図

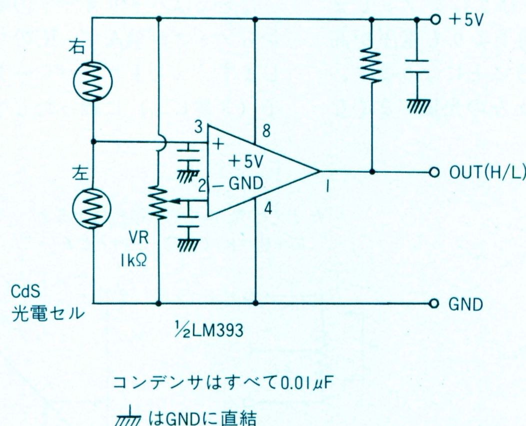


図3　CdSの構造図

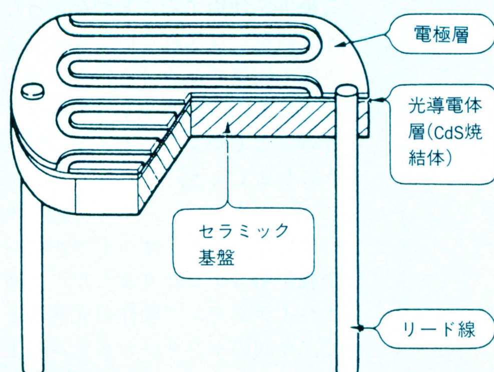
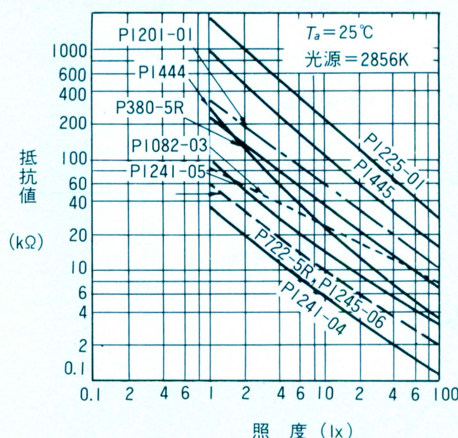


図4　CdSの照度－抵抗値特性の例



からの出力電圧も変わるということになります。

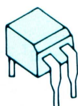
可変抵抗の代わりに今回のCdS光電セルを使用すると，光量の変化がCdS光電セルの抵抗値の変化となり，その結果分圧器の出力電圧（コンパレータの入力電圧）の変化となって検出できるのです。

今回のセンサー部分では，入ってくる光量の絶対値をアナログ的に読み取ってA/D変換する必要はなく，光量の大小だけ判断すればよいので，前回使用したフォトダイオードとオペアンプとA/Dコンバータという高級な組み合わせにする必要はなく，より簡単なCdS光電セルとコンパレータを使ってみたのです。

では，もう少し具体的にCdS光電セル付近の動作を追ってみましょう。まず，パトリオット本体が光源に対して左に向きすぎているとします。すると，左右2個のCdS光電セルのうち右のほうが光量が多い状態になります。光電セルは入ってくる光量が多いと抵抗値が下がりますから，右の光電セルのほうが抵抗値が低くなり，その結果＋端子の入力電圧は高いほうにシフトします。＋端子のほうが－端子よりも電圧が高いので，出力はHということになります。

逆に右に向きすぎると左の光量が多くなり，左の光電セルの抵抗値が下がって＋端子の電圧は低いほうにシフトし，結果として出力はLになるわけです。この出力端子はジョイスティックポートに直結されていて，コンピュータはこの端子のH・Lを逐次読み取ることで，パトリオットの向いている方向についてのデータを得るのです。

以上，電圧の変化を基準電圧との大小の比較で判断するコンパレータの動作は極めて単純明快で，それだけに応用範囲も広くなっています。今後もまた使用する機会があるかもしれないので，ぜひ使い方をマスターしておいてください。



バイパスコンデンサ

2，3，8番ピンのところに入っている0.01μFのコンデンサについて少し注意をしておきましょう。このコンデンサはバイパスコンデンサ（通称パスコン）と呼ばれ，ノイズによる誤動作を防ぐためのものです。実際このコンデンサがないとパトリオットは正常に光源を自動追尾してくれません。

たとえばパトリオットの駆動用モーターからノイズが飛んで，ICのピンに乗ったとします。もしもコンパレータの入力電圧端子（3番ピン）に載ったとすると，そのノイズの瞬間に入力端子にはノイズの電圧がかかりますから，実際に光電セルの状態とまったく関係ない電圧値でコンパレータが動作してしまいます。



その結果，パトリオットはノイズによって光源の位置を見失ってしまうことになるのです。ところが，ノイズというのはかなり瞬間的に電圧が変動するのに対し，CdS光電セルによる分圧器からの出力は変化が緩やかです。

コンデンサというものは定常的な直流成分は通しませんが，ノイズのような高周波の交流成分は通してしまう性質があります。したがって，コンパレータの3番ピンに乗ったノイズはこのバイパスコンデンサを通ってGNDに逃げていき，光電セルからの出力はコンデンサを通らずにほとんどそのままコンパレータに入力されていくことになります（図6）。

そのほかの端子についても同様に，ノイズだけGNDに逃がしてやるためにコンデンサを入れてあるのです。直流モーターからはかなりのノイズが出ており，しかもこの光センサー回路はパトリオット本体のモーターのすぐそばに取り付けられるため，バイパスコンデンサを入れておかないと100%誤動作します。

これまで製作してきた回路はあまり過酷な条件で使用することがなかったので，ノイズ対策には触れないですんできましたが，実際の回路では設計そのものは間違えていなくても，実際問題としてトラブルに悩まされることがよくありますので注意するようにしましょう。

＊　＊　＊

ついに，パトリオットマル秘システムの全貌が明らかになりました。次回はとうとうハイテクタンク製作の完成編ということで，今回のセンサーシステムの工作実習とあわせて，自動追尾制御プログラムを完成させます。ぜひ最後まで頑張ってついてきてください。

図5　分圧器

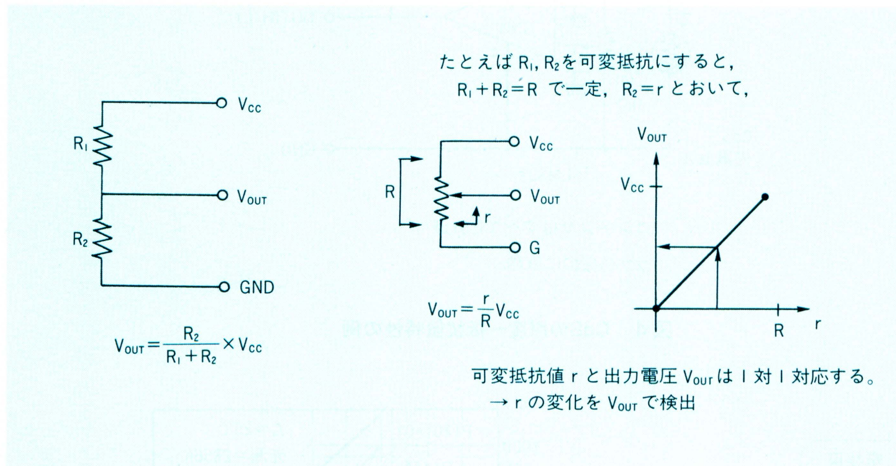
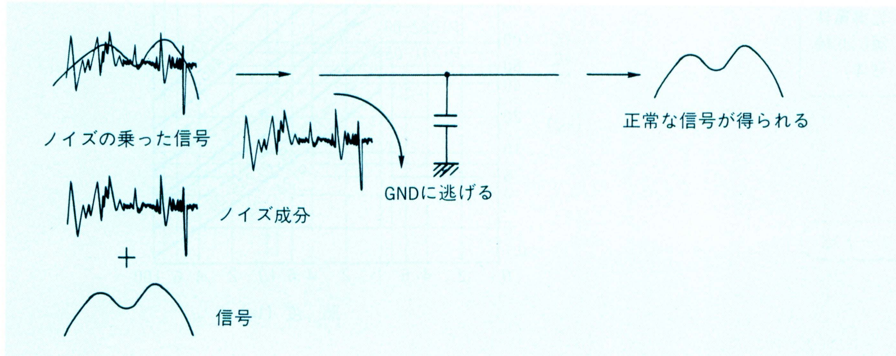


図6　バイパスコンデンサの役目



F-CARD GTをいじくる

Ogikubo Kei 荻窪 圭

11月号の新製品紹介で，金子氏が「F-CARD GT」のレビューを行いました。今回は，その「F-CARD GT」を異なった角度から斬ってみたいと思います。どういう評価が下るでしょうか。

なんか，「だって，地球は丸いんだもん」っていう気分だな。

まあ，諸般の事情もあって（どういう事情かということを説明する機会はないと思いますが），今回も日記形式で書くことをお許しください（これが景山民夫のとある連載のパロディであることなんて，どれだけの人にわかるのであろうか）。

* * *

10月某日（晴れ）

新機種発売にあわせ，アップルコンピュータジャパンはMacintosh LCを値下げした。メモリ2Mバイトでハードディスクなしのいちばん安いモデルが268,000円である。CPUは68020の16MHz。データバスは16ビットとはいえ，ハードディスクがないとまったく使えないとはいえ，RAMが2Mバイトではたいしたことはできないとはいえ，安くなったものだ。ところで，X68000 XVIって，いくらだったっけ。

* * *

10月某日（雨）

先月，金子氏がレビューした「F-CARD GT」を受け取る。MS-DOS版もあまり話題にはならなかった代物なので，期待してはいなかったが，もしかしたら，面白いソフトかもしれない。MS-DOS用のカード型データベースは玉石混淆，あまりにもたくさんありすぎるから，「安かろう悪かろう」と決めつけられてあまり吟味されていなかったのだろうか。

ちょっとスピードが遅いかな，と思ったら，シャープの「C compiler PRO-68K ver. 2.0」で開発したと書いてある。ということは，GNU Cでコンパイルしたら，もっとコンパクトで，もっと速いソフトになっていたのか。ムムム。そもそもTAKERUで発売されているソフトであるから，普通のパッケージソフトよりバージョンアップのコストはかからないはずである。期待できるか。

* * *

10月某日（台風）

私がソフトを評価するときの基準のひとつに「志が高いかどうか」というのがある。いくら完成度が高いソフトでも「志」（こころざし）が低ければ，あまり高い評価はしたくないし，「志」が高ければ多少バグがあっても（もちろん，ファイル管理やそのソフトの売りの部分などにバグが残っているものは評価外だが），洗練されていなくても評価対象になる。

「志」が低いソフトは完成した時点でもう進歩がなくなっている。低いレベルで完結しているから，期待しようがない。ユーザーの要求は常に多様多彩であり，それがソフトウェアに進歩を促す（と，いいな，と思っている）。

「志」が高いソフトというのは，「バグを取れば」，あるいは「次のバージョンになれば」絶対によくなるというのがわかっている。つまり，期待できるのだ。向上しない低いレベルで完成したソフトと，向上することが見えているソフトのどちらがいいか，というと，後者に決まっている。

よくなるとわかっているものに期待しないやつはいない。仕事上で役に立てばいい，という人が多いMS-DOS用の実用ソフトの場合，「志」より完成度の高さが要求されるから，悠長なことはいってられなくて，逆に，大きく飛躍するようないいソフトがあまり育たない土壌ができている。

X68000の場合はそこまで市場も1つひとつのソフトも成熟していないから，「志」の高いソフトを応援してなんとかまともなレベルに到達してほしいと思っている。私が「Hyperword」を応援したのも，ただひとえに，完成度は低くて，遅くて，でかくても，志の高さを感じたからであり（だから，そろそろバージョンアップしてもらわなければ困るのである），かつての「Kamikaze」にもそれを感じた。



❶オムロンのブースではここに目を奪われた

ゲームでいえば，「生中継68」である。あんなに完成度が低いのに評価が高いのは，その「志」の高さがユーザーにも伝わるからだ。古いゲームでいえば，「ジェノサイド」もそうだ。

逆に，「志」の高さを感じなくて完成度も低いという，どーしようもないソフトもまた存在する。そういうソフトは，いくら待っても，レベルが見えている。

ひどいいい方だが，純粋にユーザーとしての立場からソフトウェアを眺めた場合，こういう結論に達しざるをえない。じゃあ，荻窪圭から見て「志」の低いソフトはなにか，といわれたら，いってもいいけど，とりあえず，黙っておこう。

* * *

10月某日（きつねの嫁入り）

Oh!X11月号のシャープの広告に「Press Conductor PRO-68K」ってのがあったけど，あれはなんだ。10月発売となっていながら，私はまだ見ていないぞ。今度こそ期待していいのだろうか。

* * *

10月某日（快晴）

データショウへ行った。オムロンのブースにファジィLUNAを見にいったつもり

が，ハイレグねえちゃんの豊胸を見ただけで終わってしまった（写真1）。

＊　　＊　　＊

10月某日（通り雨）

Macintoshは去年の秋から1年間で，12万台を売ったそうだ。その前は，5，6万台だったそうな。ClassicやLCの一連の低価格シリーズが登場する前はX68000より出荷台数が少なかったのに（あくまでも，日本での話だが），たった1年で追い抜いてしまったのである。うーん。X68000だって頑張らねば。

＊　　＊　　＊

10月某日

あー。日記形式っていうのは便利だけど，ちっとも本題に入らないからやめた。

F-CARDの秘かな楽しみ

本題に入る。「F-CARD GT」である。値段のわりにはけっこう面白そうだから，今回はこいつを取り上げる。X68000をDOSマシンとして使うと思って，読んでおくれ。

「F-CARD GT」についての紹介は先月，金子氏がやってくれたので省略する。簡単に使えるカード型データベースだと思えばいい。簡単なわりにはグラフが描けたりもするようだが，まあ，最近，グラフ描画くらいどのソフトでも備えているから。

「F-CARD GT」はカード型データベースである。まあこれはそれ以外の何ものでもない。ただ，一般的なカード型データベースと違うのは，次の「F-CARD GT」が項目に指定できるデータの型を見てもらえばわかる。

［文字］
［数字］
［半角］
［漢字］
［選択］
［File］
［Prog］

上から4つはまあ，いいだろう。文字形式に“なんでもOK”と“半角文字専門”と“漢字”の3つが用意されているというだけだ。日付形式がないのはご愛嬌（なのだろうか）。

5番目は「CARD PRO-68K」にはないが，一般的には常識の部類に属する，登録したデータの中から選択して入力する選択形式の型。こいつは便利である。性別の欄に，いちいち“男”とか“女”とか“かつて男”とか入力するのは面倒だから，カーソルキーで選べるようにしてしまえ，というものだ。ちなみに，「CARD PRO-68K」でもプログラムを使えば記述できるが，このくらいは標準で持っているべきだ。

問題は最後の2つである。これが面白いのだ。

アルバムを作ろう（笑）

［File］と［Prog］のデータ型は，広告でもうたっている「プラットフォーム機能」を実現するものである。

まずは［File］型。ここにはファイル名を書くことができる。すると，指定したファイルがデータベースの一部としてアクセスできるようになるのである。

具体的には，各フィールドの［File］型の欄にファイル名を書いておく。

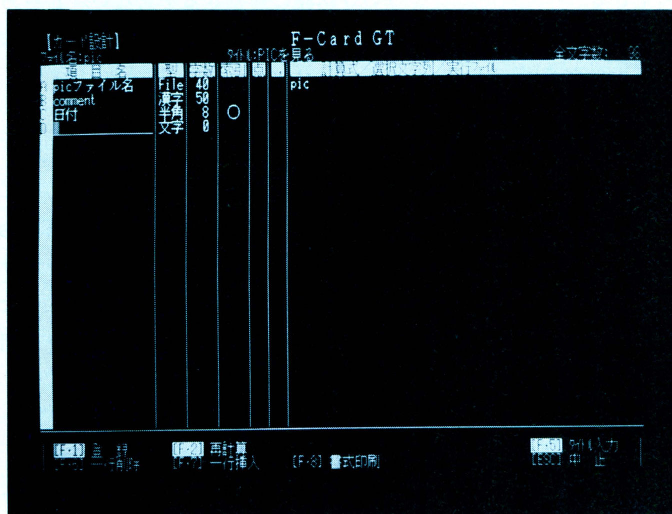
データベースを参照している状態で，その欄（フィールド）にカーソルを移動し，そこでSHIFT＋RETURNとやる。すると，そのファイルがぴよんと参照できるのだ。データベース設計時に，あらかじめ，そのフィールドのファイルをどのプログラムで呼び出すかを記述しておく。そのおかげで，ファイルに応じたコマンドが起動できるのだ。参照できるのはテキストファイルだけではない。

というわけで，サンプルの設定が写真2である。起動ファイルに“PIC”が指定してあるのが見えるだろう。で，写真3のようなカード編集の画面で，ファイル名のフィールドにカーソルを合わせてSHIFT＋RETURNキーを押すと，無事，写真1のようなグラフィックが見られるわけだ。

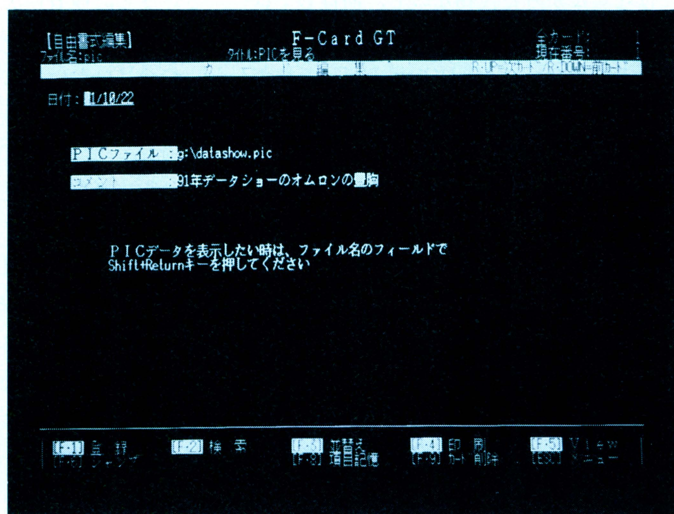
こいつは便利である。便利かどうかは別にしても，面白いものである。私はかつて，こういう機能がデータベースにほしい！と書いたことがあったが（あったと思う），それがこういうところで実現されてくるとは思わなかった。しかも，実行プログラムまで指定できるとは。

サンプルにはテキストデータを中心に，ログの管理などがついているが，応用はいくらでも効く。写真3の作りにしてしまえば，アルバムでもなんでもこいだし，実行ファイルをMicroEMACSにしてテキストファイルを管理したりもできる。

記述できるのはファイル名をお尻につけ



❷PICを呼び出すフォームの書式



❸写真①を呼び出すための画面

て起動できるプログラムであり，「F-CARD GT」が常駐していても起動できるだけのコンパクトなもの（とはいえ，「F-CARD GT」の常駐量は25Kバイト程度らしいから，2Mバイトを装備していればまあ大丈夫だろう）ならなんでもOKである。あんまりでかいプログラムは立ち上がるのに時間がかかるので，フリーウェアなどのコンパクトなものがいいだろう。

オプション付き起動も可能だ。

たとえば項目Cの内容をオプションとして，起動したいプログラムがMIとすると，

MI %C

って書けばいいのである。それでもって，項目Cを選択型にしておけば，いちいち手で打ち込まなくても，“選択”でOKだ。

ただし，“COPY ファイル名 OPM”のようなものは，[File]型では記述できないのが残念だ。が，抜け道はいくらでもあって，起動ファイルには，HISTORY.Xの“Alias”に登録した名前も使えるのだ。つまり，PLAYという名前で，

COPY %.OPM OPM

というエイリアスが登録してあれば，実行プログラムの欄にPLAYって書いてやればいいのである。まあ，バッチファイルを作っておくのも手だな。

なお，ファイル名であるが，フルパスで書いておくことを勧める。そのファイルのあるディレクトリから「F-CARD GT」を立ち上げればパス指定はいらないのだが，世の中そう甘くはないはずだ。

ちなみに，テキストデータならF5キーを押せばそのまま表示できる。グラフィックデータ（GL3などのベタファイル表示だけど）とグラフデータ表示用のプログラムはついてくる，と付け加えておこう。

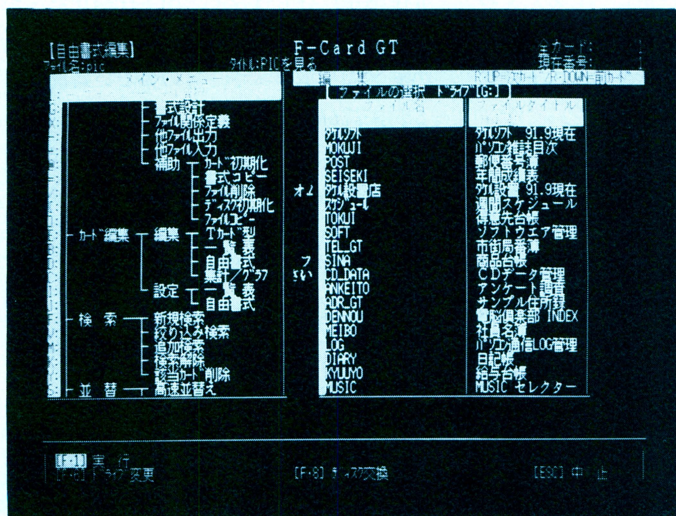
さあ，日記をつけよう(笑)。溜めたLOGを整理しよう。

シェルにしちまおう

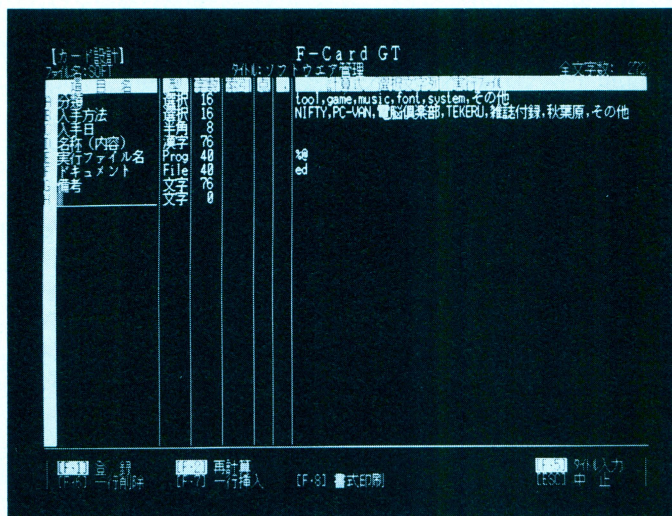
[File]型をさらに発展させたものが[Prog]型である。もうわかったと思うが，こいつは，プログラムを記述する型なのである。

「F-CARD GT」に山ほど，しかも，ようこれだけ打ち込んだ，というくらいのサンプルデータが入っているので，新しいネタを考えるのはむずかしい。なんと，サンプルデータベースは写真4のようにたくさんあるのだ。

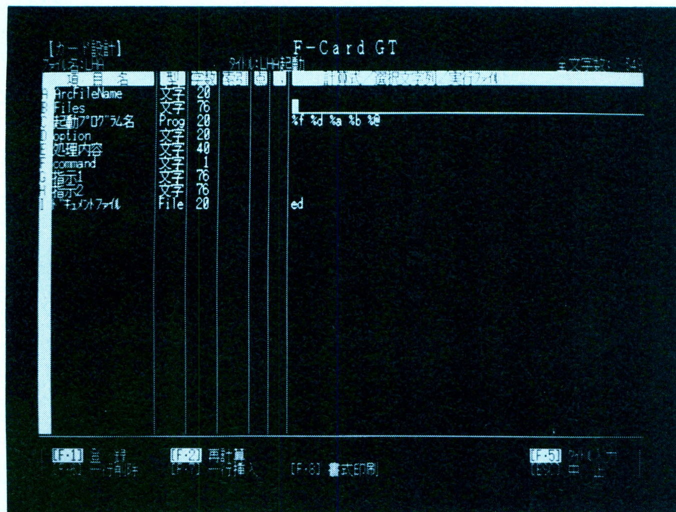
サンプルデータベースにあるソフトウェアの管理なんてのは使えそうだ。フリーウェアってのはダウンロードの味を覚えると，とたんに管理が面倒になる。実行ファイルとドキュメントを同じディレクトリにずら



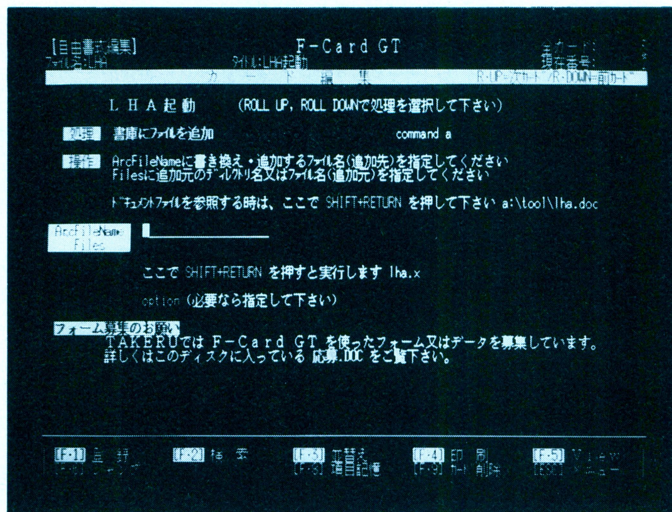
❹サンプルデータのリスト



❺ソフトウエア管理の書式設定



❻LHAを起動するフォーム



❼LHAを呼び出すカード

ずらと入れておくと，あっというまに膨れ上がって，なにがなんだったのかわからなくなる。

写真5のソフトウェア管理のフォームは，実行ファイルとドキュメントファイルをちゃんと書けるようになっているので，こいつを拡張して，ソースファイルやドキュメントとマニュアルの区別など，必要なものを登録しておけば，面倒なフリーウェアの管理も一発というわけだ。オプションをつけての起動も簡単にできるから。

写真5の解説をちょっとつけておこう。実行ファイル名の項目にある“%@”ってのは，実行時にオプションを指定するためのものである。

このソフトウェア管理フォームは便利だが，すでに溜まりに溜まったフリーウェアを登録するのが面倒なので，私のようなものぐさものにはなかなか使えないな。これはこれで困ったものだ。

だが，一度登録してしまえばデータベースであるから，目的のものをちょちょいと検索して，ちょちょいと実行するってのが（ちょちょいですむかどうかはまあ，置いておいて）可能なのだ。

そんなこんなで，実に応用範囲が広いプラットフォーム機能である。写真6のようにLHAを起動するだけのフォームもサンプルでついており，それぞれのカードには写真7のように用途別にオプションなどがつけられている。データベースということにこだわらず，こういう手もあるわけだ。

自分の環境に合わせて，なるべく最小限の手間で最大限の効果を発揮するようフォームを作れば，いちいちコマンドやオプションを入力する手間が省けるというものだ。

プラットフォーム機能の欠点

プラットフォーム機能はこのようにユニークなものであるが，2つの欠点がある。ひとつは，「F-CARD GT」の画面表示や描き変えが結構遅いので，プラットフォームとして常用するにはちょっとXVIがほしいかなということ。もうひとつは，ファイル名を入力するとき，ツリーなどのファイル一覧から選ぶことができない，ということである。手で入力しなければならないのだ。

特に，後者はもったいない。これだけ志が高い機能をつけたのだから，ちゃんとやってほしい。早急にバージョンアップで実現してもらいたいと思う。数字読み上げ機能（笑）はいらないから。

「F-CARD GT」というものは，実にDOSなソフトウェアで文句もいろいろあるが，ユニークな機能とTAKERUで8,000円というコストパフォーマンスの高さで許そう，という気にはなるな。ただ，もっとX68000らしいインタフェイスで（それには表示速度がこれではいけないが），もっと使いやすくすることは可能だ。

というところで，次のバージョンでちゃんとX68000用に作れば，かなり売れるかもしれない。なんといっても，ただのカード型データベースではないのである。いっちょ，気合いを入れてSX-WINDOW対応版ってのはどうだろうか。メニュー構造をちょっと工夫して，ちょっと速くなればユーザーは相当増えるぞ。

そんでもって，[File]型の項目をうまく使えば，データベース上から別のアプリケーションやグラフィックや音楽のウィンドウを開けるのだから。SX-WINDOWのツールやアプリケーションが増えるまでにはやってほしいのだ，と思う。

*　　*　　*

10月某日（スモッグ）

パソコン通信といえば，ASAHIパソコンネットの「電脳筒井線」がはじまった。筒井康隆氏だけではなく，山下洋輔氏だとか堀晃氏だとか中村正三郎氏だとかほうぼうの有名人が集まって，ものすごい盛況である。未読が溜まって溜まってしょうがない。なんといっても，始まって2週間で1,000以上の書き込みがあったのだ。冗談ではない。パワーはあるところにはあるものなのだ。

*　　*　　*

10月某日（眠り）

来月は「Press Conductor PRO-68K」の謎を探るぞ。

「CARD PRO-68K」パーソナルプログラム集

データベースついでに取り上げておく。「CARD PRO-68K ver.2.0」用のパーソナルプログラム集が登場した。値段は12,000円で，プログラム機能を駆使したまっとうなものから，変なものまで，いろいろと詰まっている。

昔，この連載で，「コマンドのリファレンスがわかりにくいから苦労した」と書いたわけだが，そいつがサンプルデータ集についた。マニュアルのお尻に，「コマンド＆ノウハウ」という章があるのだ。ここには，機能別のリファレンスのほか，「CARD PRO-68K」についてきたサンプルプログラムの解説が入っている。

おいおい，こういうのはちゃんと本体につけておくように，っていいたいけど，ねえ。どうでしょうか。

さて，この「パーソナルプログラム集」についているサンプルプログラムの中身を簡単に紹介しておこう。

1) 真面目な世界
   - 住所録・名刺管理
   - 電話帳
   - カレンダー
   - 週間予定表
   - Don't forget　（覚え書き）
2) お茶の間な世界
   - CD管理
   - ビデオレーベル
   - 蔵書管理
   - カロリー計算
   - ローン計算
   - 健康管理
   - 家計簿
3) 趣味の世界
   - フットボールリーグ
   - F1グランプリデータ管理
   - ゴルフオフィシャルハンディキャップ計算
   - ゴルフスコア計算
4) 色もの
   - スーパーパズ　（ソリティアの一種）
   - 15パズル
   - 易（六十四卦の易占い）
   - モグラ叩き

とまあ，いろいろあって，どれも，データベースとプログラムのソースファイルと，コンパイル済みファイルとヘルプファイルなどがセットになっているので，勝手気ままにいじるのが前提になっている。

プログラムっていってもあまりあてにしてはいけないが，なかなか面白いものではある。一覧表からの入力や，条件によってぽんぽん開くウィンドウなど，「CARD PRO-68K」っぽいものばかりなので，すでに「CARD PRO-68K」を買っていて，さあ，プログラムを書くぞ，ってな人は購入する価値はある。今回は紹介までに。

X68000用
OH YEAH!
Abe Toshimitsu
阿部 俊光

X1/turbo用
サイレント・イヴ
Sasaki Kouji
佐々木 孝司

おまけ X68000用
ジングルベル
編集部

今月はX68000とX1に1曲ずつです。季節モノってやっぱりいいですよね。ちなみに投稿するなら3カ月は先を見越さないと間に合わないかもよ。いまなら「卒業・入学」関係，「落ちたら渋谷ゼミナール」関係などを送るとちょうどいいようです。

小田和正ぢゃないぜ

X68000のOPMD用には，もうすっかりお馴染みになってしまったPRINCESS PRINCESSの「OH YEAH!」をお届けしましょう。この曲は，ソニーのカセットテープのCMでよく流れていた曲ですから，サビの「OH YEAH! 抱きしめたい～♪」ってあたりは有名ですよね。でも，せっかくのクリスマス特集（?）なのですから，同じPRI²でも「DING DONG」のほうがタイムリーだったかもしれませんが。

作ってくれたのは，1991年9月号でX1用「WHITE MANE」が掲載されている阿部君です。ちょうどその号ではPRI²も載っていたんですよね。これもなにかの縁なのかな，ちょっと不思議な気もします。

この作品はノリがいいですね。もともとの曲からしてノリはいいほうですけど，よく再現しています。決して「あのコードはGをもう少し前に出してほしかったな」なんて贅沢はいいません。間奏も含めて，なかなかよくまとまっています。ギターの音なども感心させられました。とても16歳の技とは思えません。

強いていうならば，部分的に重くなると感じるフレーズがあります。そのあたりではテンポを上げてみるとか，ちょっとした対策も考えてみてもよいでしょう。

プログラムは，前述のとおりOPMD用になっています。音色のコンフィグレーションファイルが付属していましたので，一応そちらのほうも掲載しておきます。手持ちのシステムに合わせて演奏させてください。



PRINCESS PRINCESS

阿部君はX68000ユーザーになったわけではなく，X1ユーザーです。X68000をちょっと借りた間にこの作品を作ったそうです。それから以前に阿部君から送られていたX68000用のOPMファイルをX1で鳴らせるミュージックドライバですが，ほかにもミュージックドライバが投稿されており，いろいろと検討している段階です。

これからもXシリーズ用にどんどん投稿してくださいね。

いまが聴きごろ，旬の作品

いやあ～，12月号ですねえ。12月といえばクリスマス。なにかと騒ぎたがる日本人には，切っても切れない大事な行事に成り上がりました。街ではカップルがこれみよがしに肩を並べ，めし食うところはカップルの山，東京ディズニーランドはカップルの渦。はふっ，ちょっとため息（ちなみにこの原稿を書いているのはハロウィンよりも前だったりして季節感がない）。

そんなこたぁどーでもいいんだ。今月のX1のMusic BASIC用には，辛島美登里さんの「サイレント・イヴ」をお送りしましょう。この曲はアルバム「GREEN」のなかからの選曲です。TBS系のTVドラマ，「クリスマス・イブ」の主題歌でもあったので，知っている人も多いかもしれませんね。佐々木君の投稿はちょっと古くて，以前に掲載された「NO. NEW YORK」と同時でした。まあ，せっかくのクリスマスソングだから12月号まで取っておきましょう，ということで今月掲載になりました。


辛島美登里

さて曲の話です。クリスマスっていえば前述のとおりカップルが天下を取る日なのに，当節のクリスマスソングって悲しいものが多いですよね。この「サイレント・イヴ」もそのテの曲で，美しいヴォーカルが印象的です。

この作品はピアノ6声，ヴォーカル2声で構成されています。きっとピアノの弾き語りの楽譜を参考にしたのでしょう（違ったらゴメン）。曲調からするとこのままでも十分なのですが，オーケストレイションにもこだわってほしいところですね。もしくは，ピアノにもっと気を使ってください。

ちょっと手抜きくささが残ります。音色ももう少し煮詰められそうだし。

正直なところ，辛島のお姉さまを結構気に入ってたりするんだ，私は。だから妥協は許したくないのです。

オマケは福引のあかだま級

はっはっは。今月号をなにがなんでもクリスマス特集にするために，編集室に徹夜して作りました。ものの5分もあれば入力できます。曲はいかにもの「ジングルベル」，X68000用です。サンプリングは使っていませんので，普通のシステムで演奏できます。

このプログラムには正しい遊び方があります。まず，プログラムを入力してください。間違いがないことを確認してから，おもむろにRUNしてください。このままではなんでもないでしょ。そこで，X68000のキーボードを引っこ抜き，ファンクションキーのF1，F2，F3を同時に押しながらキーボードを差し込んでみましょう。ほーら，クリスマスツリーだ。電気を暗くして，あとは好きなことをやってください。

ちなみにこのテのワザは数種類あり，「ナイトライダー」，「2進数」などが有名です。キーボードのROMに入っているお遊びですが，初代X68000からの完全コンパチビリティを保ちながらXVIまで続いているなんて，ちょっと感激ですよね。

それでは今月の名曲たちを皆さんが入力することを願いつつ，また来月。　(S.K.)

リスト1　OH YEAH!

```
  10 /*
  20 /*            『 O H   Y E A H ! 』
  30 /*
  40 /*   Princess Princess    by T.abe 1991
  50 /*
  60 dim str pd(50)[256]
  70 dim char p(255),v(4,10)
  80 char t
  90 str a[256],a1[256],a2[256]
 100 str s[10],s4[10],s8[10],s16[10],sf[40]
 110 str b[10],b4[10],b8[10],b16[10]
 120 str e[10],e8[10],e16[10],c[10],c4[10],c8[10]
 130 str da[256],de[256],db[256],ue[256],ub[256]
 140 int i,j
 150 /*
 160 m_init()
 170 m_tempo(145)
 180 for i=1 to 8
 190   m_alloc(i,7000)
 200   m_assign(i,i)
 210 next
 220 vset()
 230 mset()
 240 m_play()
 250 end
 260 /*
 270 /* trk set
 280 func trk(t)
 290   i=0
 300    while p(i)<>255
 310     m_trk(t,pd(p(i)))
 320   i=i+1
 330  endwhile
 340  return()
 350 endfunc
 360 /*
 370 /*  voice
 380 func vset()
 390 v={
 400 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 410           48, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 420 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 430           31,  4,  0,  6, 15, 14,  0,  1,  0,  0,  0,
 440           31, 31,  0,  6,  0, 18,  0,  1,  0,  0,  0,
 450           31, 31,  0,  6,  0, 23,  0,  1,  0,  0,  0,
 460           31, 31,  0,  8,  0,  0,  0,  1,  0,  0,  0}
 470 m_vset(4,v)                      :/* Guitar
 480 v={
 490 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 500           58, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 510 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 520           25, 14,  0,  2,  0, 22,  0,  1,  0,  0,  0,
 530           23, 12,  0,  2, 15, 40,  0,  3,  0,  0,  0,
 540           23,  0,  0,  2,  0, 28,  0,  1,  0,  0,  0,
 550           18,  0,  0,  7,  0,  0,  0,  1,  0,  0,  0}
 560 m_vset(5,v)                      :/* Back
 570 v={
 580 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 590           58, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 600 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 610           30,  2,  0,  5,  1, 35,  0,  1,  0,  0,  0,
 620           31,  6,  1,  8,  3, 24,  0,  5,  7,  0,  0,
 630           28,  3,  0,  6,  1, 47,  0,  1,  0,  0,  0,
 640           31,  4,  1,  6,  0,  0,  0,  1,  4,  0,  0}
 650 m_vset(6,v)                      :/* Keyboard
 660 v={
 670 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 680            8, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 690 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 700           31, 18,  0,  6,  2, 33,  0, 10,  0,  0,  0,
 710           31, 14,  4,  6,  2, 41,  0,  0,  7,  0,  0,
 720           31, 10,  4,  6,  2, 17,  1,  0,  3,  0,  0,
 730           31, 10,  3,  6,  2,  0,  1,  0,  0,  0,  0}
 740 m_vset(8,v)                      :/* Bass
 750 v={
 760 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 770           58, 15,  2,  1,200, 99,  0,  2,  0,  3,  0,
 780 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 790           31,  4,  1,  6,  1, 25,  0,  1,  6,  0,  0,
 800           31, 15,  1,  8,  5, 35,  0,  8,  0,  0,  0,
 810           31, 31,  1,  3,  0, 34,  0,  1,  5,  0,  0,
 820           31, 31,  1, 10,  0,  0,  0,  2,  0,  0,  0}
 830 m_vset(9,v)                      :/* Vocal
 840 v={
 850 /*        AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 860           61, 15,  2,  1,200, 99,  0,  2,  0,  3,  0,
 870 /*        AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
 880           17, 15,  0, 10,  0, 25,  0,  1,  0,  0,  0,
 890           20,  8,  0, 15,  0,  0,  0,  1,  3,  0,  0,
 900           20,  8,  7, 15,  0,  2,  0,  1,  1,  0,  0,
 910           10,  0,  0, 15,  0,  0,  0,  1,  3,  0,  0}
 920 m_vset(10,v)                     :/* Brass
 930 endfunc
 940 /*
 950 /*  mml data
 960 func mset()
 970 s="y2,15"
 980 s4=s+"r4"
 990 s8=s+"r8"
1000 s16=s+"r16"
1010 sf=s16+s16+s16+s16
1020 b="y2,23"
1030 b4=b+"r4"
1040 b8=b+"r8"
1050 b16=b+"r16"
1060 e="y2,29"
1070 e8=e+"r8"
1080 e16=e+"r16"
1090 c="y2,5"
1100 c4=c+"r4"
1110 c8=c+"r8"
1120 da="@L2g+&g&f+&f&e&d+&d&c+&c&>b&a+&a<L8"
1130 de="@L2d+&d&c+&c&>b&a+&a&g+&g&f+&f&e<L8"
1140 db="@L2a+&a&g+&g&f+&f&e&d+&d&c+&c&>b<L8"
1150 ue="@L3f&f+&g&g+&a&a+&b&bL8"
1160 ub="@L2<c&c+&d&d+&e&f&f+&g&g+&a&a+&b>L8"
1170 /* vocal
1180 a1="a4a4a4a4ac+der>aaa<d4dddedd&>aa"
1190 a2="a4r2af+aaaa4.r4b4b4beee<c4c4c>ba"
1200 pd(0)="@9  @v127  q7 L8 o4 y48,20
1210 pd(1)="r1r1"
1220 pd(2)="r1r1r1r2ref+e"
1230 pd(3)=a1+"4.r<ef+e"
1240 pd(4)=a1+"a2r4<"
1250 pd(5)=a1+"a4r2<"
1260 pd(6)="r4g+g+g+f+g+a&ee4.r2 r4g+g+g+bba&ee4.r2"
1270 pd(7)="r4g+g+g+f+g+g+ad4.rddd+&d+2rd+d+e&"
1280 pd(8)="e1g+1b2.r4<c4c4c>baa&"
1290 pd(9)=a2+"a&"
1300 pd(10)=a2+"<c+&>"
1310 pd(11)="a1r1"
1320 pd(12)="r1r2ref+e"
1330 pd(13)="a1&a1r1r1"
1340 pd(14)="r2.af+aaaa4.r4 b4b4beee<c4c4c>baa4.r2 af+aaaa4.r4
b4b4bee4<rc4.c>b&a<c+>"
1350 /*
1360 p={ 0,1,1,1,1,2, 3,4,3,5, 6,7,8, 9,9,9,10, 11,1,1,12,
1370                  3,4,3,5, 6,7,8, 9,9,9,10, 13,1,1,14,
1380                                  9,9,9,10, 11,1,1,1 ,255}
1390 trk(1)
1400 /*
1410 /* guitar 1
1420 pd(0)="@4  v13y49,20 q8 L8 o3 p3"
1430 pd(1)="araraaaraaaar<dd4>"
1440 pd(2)="b2.&bg1f+&f+2.<e&g+&g+1>"
1450 pd(3)="g+&aaag+&aaa g+&aag+&aaaa f+&gggf+&ggg <c+&ddc+&ddd
d>"
1460 pd(4)="p1>eb<e4d&e>e4 a<<c+c+4d4c+q2c+16c+16q8> >eb<e4d&e>
e4 a<<c+d4dd4c+>"
1470 pd(5)=">eb<e4d&e>e<<q5d16d16q8d2.dq2d16d16q8>b2.&b&"+ub
1480 pd(6)="p3<erereee&"+de+" eeeeree&"+de+">"
1490 pd(7)="v11q5aaaq2a16a16q5aaa4 aaaq2a16a16q5aaa4 bbbq2b16b1
6q5bbb4 aaaq2a16a16q5ggggq8v13"
1500 pd(8)="<e&fffe&fff e&ffe&ffff f+&gggf+&ggg f+&ggf+&gggg>"
1510 pd(9)="<arr2.r1r1r1>aaarr1r1r1rege>"
```

```
1520 pd(10)="araraaaraaaar2"
1530 /*
1540 p=( 0,1,1,1,1,2, 3,3,3,3, 4,5,6,6, 7,7,7,7, 1,1,1,1,
1550                  3,3,3,3, 4,5,6,6, 7,7,7,7, 8,8,9,
1560                                    7,7,7,7, 1,1,1,10,255)
1570 trk(2)
1580 /*
1590 /*  guitar 2
1600 pd(0)="@4  v13y50,40q8 L8 o3 p3"
1610 pd(1)="erereeereeeeraa4"
1620 pd(2)="f+2.&f+d1>b&b2.<r4r1"
1630 pd(3)="d+&eeed+&eee d+&eed+&eeee c+&dddc+&ddd g+&aag+&aaaa
"
1640 pd(4)="p1>eb<e4d&e>e4 a<ee4f+4eq2e16e16q8 >eb<e4d&e>e4 a<e
f+4f+f+4e"
1650 pd(5)=">eb<e4d&e>e<q5a16a16q8a2.aq2a16a16q8f+2.&f+&@L2g&g+
&a&a+&b&<c&c+&d&d+&e&f&f+>L8"
1660 pd(6)="p3brbrbbb&"+db+" bbbbrbb&"+db
1670 pd(7)="v11q5eeeq2e16e16q5eee4 f+f+f+q2f+16f+16q5f+f+f+4 g+
g+g+q2g+16g+16q5g+g+g+4 f+f+f+q2f+16f+16q5ddddq8v13"
1680 pd(8)="b&<ccc>b&<ccc> b&<cc>b&<cccc c+&dddc+&ddd c+&ddc+&d
ddd>"
1690 pd(9)="<err2.r1r1r1>eeerr1r1r1rege"
1700 pd(10)="erereeereeeer2"
1710 /*
1720 trk(3)
1730 /*
1740 /*  keyboard 1
1750 pd(0)="@6  v13y51,20 q8 L8 o4 p3"
1760 pd(1)="r1r1"
1770 pd(2)="r1r1r1req4gq8rf+q4gq8r4"
1780 pd(3)="r1r1r1f+a@L2e&@L22f+L8ec>baa&"
1790 pd(4)="a4r2.r1r1r1<"
1800 pd(5)="@6p2rbbrbbr4<c+r4drc+4.>"
1810 pd(6)="rbbrbbr4<ddrdrdd4>f+1"
1820 pd(7)="p3brbrbbbrbbbbrbbr"
1830 pd(8)="@10v13o3aa<cc+eq4eq8r4 ddff+aq4ar4 r<er4q8eq4er4q8f
+2gf+e>>a&"
1840 pd(9)="aa<cc+eq4eq8r4 ddff+aq4ar4 q8r2r<ed+er2g2"
1850 pd(10)="r4.q4eq8cq4c+r4 q8rgf+ef+q4gq8r4 r2r>ed+ed+ed+ed+e
f+e"
1860 pd(11)="gf+e>a4<a4rr4<c>baq4aq8r4r2<eed+d&d2g2"
1870 pd(12)="r1r4.q4eq8gf+ga"
1880 pd(13)="rd4g4a4.&a1>"
1890 pd(14)="r1r2ra<c>@L4a+&@L44bL8<d>aeggb<d4eggee>b&"+db
1900 pd(15)="r4.@L8cdfL8aay59,80y60,80a+a+ a+a+16&bb16<y59,64y6
0,64c4>g16d16y59,0y60,0gg b<q4degq8y59,64y60,64b2.&b8.y59,80y60,
80a16<c&>b4.y59,32y60,32"
1910 pd(16)="arr2.r1r1r1 aaarr1r1r1r>ege"
1920 pd(17)="a1&a1"
1930 pd(18)="<a1aaaar2"
1940 pd(19)="r1r1"
1950 /*
1960 p=( 0,1,1,1,1,1,1,
1970      1,1,1,1,1,1,1,1, 5,5,6,7,7, 8,9,10,11, 1,12,19,13,
1980      1,1,2,3,4,       5,5,6,7,7, 8,9,10,11, 14,15,16,
1990                                  8,9,10,11, 1,1,17,18,255)
2000 trk(4)
2010 /*
2020 /*  keyboard 2
2030 pd(0)="@6  v13y52,40 q8 L8 o4 p3"
2040 pd(5)="@6p2rg+g+rg+g+r4arrara4."
2050 pd(6)="rg+g+rg+g+r4aararaa4d+1"
2060 pd(7)="p3g+rg+rg+g+g+rg+g+g+g+rg+g+r"
2070 pd(8)="@10v13o3aa<cc+c+q4c+q8r4 ddff+f+q4f+r4 rg+r4q8gq4g+
r4q8f+2gf+e>a&"
2080 pd(9)="aa<cc+c+q4c+q8r4 ddff+f+q4f+r4 q8r2rbbb r2<d2>
2090 pd(10)="r4.q4eq8cq4c+r4 q8rgf+ef+q4gq8r4 r2red+ed+ed+ed+ef
+e"
2100 pd(11)="gf+e>a4<a4rr4 c>baq4aq8r4r2<eed+d&d2g2"
2110 pd(13)="a1&a1<"
2120 pd(16)=">arr2.r1r1r1aaarr1r1r1rege"
2130 pd(17)=">a1&a1"
2140 pd(19)="r1r2c>baa&"
2150 /*
2160 trk(5)
2170 /*
2180 /*  Bass
2190 a="aa<c&c+ec&c+eddf&f+af&f+a>eeg&g+bg&g+b<ddd"
2200 pd(0)="@8  v13   q7 L8 o3 p3 y53,20
2210 pd(1)="araraaaraaaaraa&"+da
2220 pd(2)="araraaaraaaar<aa4>"
2230 pd(3)="b2.&bg1e&e1r<<d+&e>a+&be4.>"
2240 pd(4)="g+&aaag+&aaa g+&aag+&aaaa f+&gggf+&ggg <c+&ddc+&ddd
d>"
2250 pd(5)="<d+&eeed+&eee> g+&aag+&a<g+&aa d+&eeed+&eee> g+&aag
+&aaaa"
2260 pd(6)="<d+&eeed+&eeec+&ddc+&ddd&"+de+">bbbbbb4"+ub
2270 pd(7)="<erereee&"+de+" >eeeeree&"+ue+"<erereee&"+ue+"<eeee
ee16&@L3f&f+&g&g+g+8&@L2g&f+&f&e&d+&d&c+&c&>b&a+&a&g+>L8"
2280  pd(8)=a+da+">gf+ea"
2290  pd(9)=a+"4gf+ed>"
2300 pd(10)=a+da+">gf+gg+"
2310 pd(11)=a+"4gggg>"
2320 pd(12)="araraaaraaa<aa>a&"+da+"r"
2330 pd(13)="e&fffe&fff e&ff<e&ffff> f+&gggf+&ggg f+&gg<f+&gggg
>"
2340 pd(14)="a&"+da+"r2.r1r1r1aaa&"+da+"r2r1r1 r16<<<@L2c8&>b&a
+&a&g+&g&f+"+de+">e4&"+de+">re"
2350 pd(15)="araraaa&@L2a+&b&<c&c+&d&d+&e&f&f+&g&g+&aL8aaaa32&"
+da+"r4.r16."
2360 /*
2370 p=( 0,1,2,1,1,3,
2380      4,4,4,4, 5,6,7, 8,9,10,11, 1,1,1,12,
2390      4,4,4,4, 5,6,7, 8,9,10,11, 13,13,14,
2400                      8,9,10,11, 1,1,1,15     ,255)
2410 trk(6)
2420 /*
2430 /*  Drums
2440 pd(0)="L8y3,3"
2450 pd(1)=s4+s4+s8+s8+s4+s8+s8+s8+s4+c8+c4
2460 pd(2)=s4+s4+s8+s8+s8+b16+b16+s8+s8+s8+s4+c8+c4
2470 pd(3)=e8+e8+s8+e16+s16:pd(3)=pd(3)+pd(3)+pd(3)+e8+s8+c8+e8
2480 a=e8+e8+s8+e16+s16:pd(4)=a+e8+e8+s8+e8+a+e8+e8+sf
2490 pd(5)="y2,3r"+b8+s8+e16+s16+e8+e8+s8+"y2,3r4"+b8+s8+e16+s1
6+e8+e8+s8+"y2,3rr1r2r"+b8+s+"@L4r"+s+"@L44rL8"
2500 pd(6)=b4+s4+b4+s4+b4+s8+b4+b8+s4
2510 pd(7)=b4+s4+b4+s4+b4+s8+b8+"y2,28r8."+s16+s8+"y2,28r8"
2520 pd(8)=b4+s4+b4+s4+b4+s8+b8+"y2,28r8."+s16+s8+s16+s16
2530 pd(9)="@L8y3,1y2,28ry3,3y2,28ry3,2y2,29ry3,3":pd(9)=b4+s8+
pd(9)+b8+pd(9)+b8+b8+"L8"
2540 pd(10)=s4+s4+s8+s8+s4+b8+s16+s16+sf+sf+sf
2550 pd(11)=b4+s4+b8+b8+s4
2560 pd(12)=b4+s4+b8+b8+s8+b8
2570 pd(13)="y2,3ry2,3r4."+b4+s4
2580 pd(14)=b4+s4+s+"@L4r"+s+"@L20rL8"+s8+s8+b8
2590 pd(15)="y2,3r4"+s8+"."+s16+b8+b8+s4
2600 pd(16)=b4+s8+"."+s16+sf+sf
2610 pd(17)=a+a+a+e8+e8+c4
2620 pd(18)=a+a+e8+e8+s8+e8+e8+e8+c4
2630 pd(19)=a+a+a+sf+sf
2640 pd(20)="r"+b8+s8+b8+c8+"."+s16+sf
2650 a=b8+b8+s8+b8:pd(21)=a+a+a+"r"+b8+s8+b8
2660 pd(22)=a+a+a+"y2,28r8."+s16+s8+s16+s16
2670 pd(23)="y2,3r"+b8+s4+b8+b8+s4
2680 pd(24)=b4+s4+b8+b8+s8+s8+"r8."+s16+sf+sf+s16+s16+c8
2690 pd(25)=s4+s4+s8+s8+s4+s8+s8+s8+s4+c8+c8+b16+b16
2700 pd(26)=s8+b16+b16+s8+b16+b16+s8+s8+s8+b16+b16+s8+s8+s8+s8+
b8+c8+c8+b8
2710 pd(27)=c4+c4+c8+c8+c4+c8+c8+c8+c4+c8+c4
2720 pd(28)=s4+s4+s8+s8+s4+s8+s8+s8+c8
2730 /*
2740 p=( 0,1,2,3,4,5,6,6,6,7,6,6,6,8,6,6,6,9,1,10,
2750     11,11,11,12,13,11,11,14,11,11,11,12,15,11,11,16,
2760     17,18,18,19,
2770     6,6,6,7,6,6,6,8,6,6,6,9,1,10,
2780     11,11,11,12,13,11,11,14,11,11,11,12,15,11,11,20,
2790     21,21,21,22,11,11,11,11,23,11,24,
2800     11,11,11,12,13,11,11,14,11,11,11,12,15,11,11,20,
2810     25,26,27,28, 255)
2820 trk(7)
2830 /*
2840 /*  Back
2850 a="@9q8v15r4@L2a+&b&<@L44c+>L8a2v14q7"
2860 pd(0)="    q8 L4 o4 p3 y55,40"
2870 pd(1)="r1r1r1r1"
2880 pd(2)="r1r1r1r2@5q8v14L4eb"
2890 pd(3)="g+2b2a2.r g+2b2areb g+2b2a2.r f+1 e1e1g+2.rr1"
2900 pd(4)=a+"r1r1r1"
2910 pd(5)="y2,3"+a+"f+f+f+f+4.r4g+4g+4g+eeea4a4af+ee&"
2920 pd(6)="y2,3"+a+"f+f+f+f+4.r4g+4g+4g+eeea4a4af+ea&"
2930 pd(7)="e1r1r1r1"
2940 pd(8)="y2,3f1r1r1r4.y2,3r4y2,3r4. y2,3r1r1r1r1"
2950 pd(9)="y2,3"
2960 /*
2970 p=(0, 1,1,1,1,1,1,2,3,4,4,5,6,9,7,1,
2980     1,1,1,2,3,4,4,5,6,8, 9,4,9,4, 4,4,5,6, 7,1,255)
2990 trk(8)
3000 /*
3010 endfunc
```

リスト2 OH YEAH! コンフィグファイル

```
03 = a:¥pcmdata¥crash1.PCM
05 = a:¥pcmdata¥sn_cr1.PCM
15 = a:¥pcmdata¥sn2_5.PCM
23 = a:¥pcmdata¥kick3.PCM
28 = a:¥pcmdata¥etom1_1.PCM
29 = a:¥pcmdata¥etom1_2.PCM
```


リスト3 サイレント・イヴ

```
10  '+-------------------------------------------------+
20  '|                                                 |
30  '|        「Silent Eve」              Version 1.50  |
40  '|                                                 |
50  '|         Music    By  M.Karashima                |
60  '|        Program   By  K.SASAKI(VIP000 CHACHA)     |
70  '|                                                 |
80  '|
90  '|      Special thanks to VIP ROOM                 |
100 '|                           Feel Dizzy            |
110 '|                           BALLADE SPORTS CR-X   |
120 '|                                                 |
130 '|           1990/12/25 - 1991/01/05               |
140 '|                                                 |
150 '+-------------------------------------------------+
160 '
```

```
170 R=0 : SCREEN : CLS 1 : GOSUB 2710
180 TEMPO0 : PLAY"T84"
190 GOTO270
200 '--<< PLAY >>--
210 LABEL"!"
220 'POKE&hFEF0,1,1,1,1,1,1,1,1
230 PLAY    A$;:PLAY":"+B$;:PLAY":"+C$;:PLAY":"+D$;
240 PLAY":"+E$;:PLAY":"+F$;:PLAY":"+G$;:PLAY":"+H$
250 RETURN
260 '--<< MML DATA >>--
270 A$="I03 @V117 Q7 L8 K7"     ' Vocal
280 B$="I03 @V106 Q7 L8 K3"     ' Vocal
290 C$="I01 @V118 Q8 L8"        ' Piano
300 D$="I01 @V118 Q8 L8"        ' Piano
310 E$="I01 @V118 Q8 L8"        ' Piano
320 F$="I01 @V118 Q8 L8"        ' Piano
330 G$="I01 @V118 Q8 L8"        ' Piano
340 H$="I01 @V118 Q8 L8"        ' Piano
350 "!"
360 A$="R1             R1"
370 B$="R1             R1"
380 C$="R1             R1"
390 D$="O5G+A16B@156 F+G+16A@156"
400 E$="R2O5EG+4.     R2D+F+4."
410 F$="R1             R1"
420 G$="R1             R1"
430 H$="O4I02E1        D+1"
440 "!"
450 D$="EC+F+D+E4D+<B16>I02C+16&  C+1"
460 E$="O4G+4G+4>C+4<F+R16I02A16& A1"
470 F$="R2O4G+4R.I02E16&          E1"
480 H$="I01C+4<B4A4G+.I02F+16&    F+1&"
490 "!"
500 A$="R1"
510 B$="R1"
520 C$="R1"
530 D$="O5G+@4R@4R@4O6E@179I01
540 E$="R@4O5A@4I01
550 F$="R@4R@4O6C+@4I01
560 G$="R1"
570 H$="F+1"
580 "!"
590 '--<< A >>--
600 A$="_4O4G+4.A16G+16G+4R.G+16 G+4.A16F+@60C+D+"
610 C$="R1                       R1"
620 D$="O5E<G+>C+4E<G+>C4        E<G+B4>F+<A+>C+4"
630 E$="R1                       R1"
640 F$="R1                       R1"
650 G$="R1                       R1"
660 H$="O4I01C+2C2               <B2A+2"
670 B$="R@6"+A$+"@18&"
680 "!"
690 A$="E4EG+F+4D+<B&  B2R"
700 D$="E<G+>C+4D+4<B4 B<B>D+4>C+4C4"
710 E$="R2O5C+4R4      <G+RR4G+4G+4"
720 F$="R1             O4D+RR4F+4F+4"
730 H$="A2B2           E2G+2"
740 B$="D+@6"+A$+"@90":A$=A$+"@96"
750 "!"
760 A$="O4G+4RA16G+16G+4R.G+16 G+4.>C+16<F+@60C+8D+"
770 D$="O4G+>C+G+4<G+>CG+4     <G+B>G+4<F+A+>F+4"
780 E$="R1                     R1"
790 F$="R1                     R1"
800 G$="O4E2E2                 E2C+2"
810 H$="O4D+4C+4C2             <B2A+2"
820 B$="R@6"+A$+"@18&"
830 "!"
840 A$="E4RD+16E16F+@32D+@32<B@32 >G+2R4.<B"
850 D$="O5R@6E@90R@6E@90          R@6I02G+@186"
860 E$="O5R@3C+@93R@3C+@93        R@3I02D+@189"
870 F$="O4G+@96A@96               I02B@191I01"
880 G$="R1                        R1"
890 H$="O2A>EG+4<B>F+A4           EB>D+<B>G+2"
900 B$="D+@6"+A$+"@18&"
910 "!"
920 ' --<< B >>--
930 A$="A2B>C+D+A16G+16& G+4R4<AB>C+G+16F+"
940 C$="R1               R1"
950 D$="I01O4A2A2        G+2G+2"
960 E$="I01O4E2E2        E2E2"
970 F$="O4C+2C+2         <B2R2"
980 G$="R1               R1"
990 H$="O2F+>C+A4<B>F+A4 <EB>G+4EB>C+4"
1000 B$="B@6"+A$+"@6&":A$=A$+"16&"
1010 "!"
1020 A$="F+4R4.F+F+D+            F+E16E@60R4.<~2B"
1030 D$="F+2F+2                  G+2F+4G+4"
1040 E$="C+2D+2                  E2D+4E4"
1050 F$="<A+2>C2                 C+2C+4C+4"
1060 H$="O2D+A+>F+>C+<<G+>D+G+4 C+G+>C+E<D+4E4"
1070 B$="F+@6&"+A$+"@18&"
1080 "!"
1090 A$=">C+4R4D+EF+>C+ <B4.BE4R~2E16F+"
1100 D$="G+2F+2         F+2E2"
1110 E$="E2D+2          D+2C+2"
1120 F$="C+2C+4<B4      B2G+2"
1130 H$="<A>EG+4<B4A4   G+>D+F+4C+4.<B"
1140 B$="B@6"+A$+"@6":A$=A$+"16"
1150 "!"
1160 A$="G+4.G+G+BAG+&     G+1             R2R<B>EF+"
1170 D$="G+2G+2            G+4.G+4C+G+D+ G+2.E4"
1180 E$="F+2R2             D+4.D+4R4.     C2.C+4"
1190 F$=">C+2R2            C+4.C+4R4.     R2.A4"
1200 H$="A+>F+A+>C+G+4C+4 <<G+>D+G+2.    <G+>D+G+B>G+4<<B4"
1210 B$="F+@6"+A$+"@18&"
1220 "!"
1230 ' --<< C >>--
1240 A$="G+A16B@60R<B>EF+ G+A16B@60R<B>EF+"
1250 C$="R1               R1"
1260 D$="I02O4B1          B1"
1270 E$="RO4EF+EBEF+E     RD+F+D+BD+F+D+"
1280 F$="R1               R1",
1290 G$="R1               R1"
1300 H$="O3E4.EE4.E       D+4.D+D+4.D+"
1310 B$="F+@6"+A$+"@18&"
1320 "!"
1330 A$="G+A16B.AG+AB>C+& C+<A4.R_2F+F+G+"
1340 D$="B1               A2.R4"
1350 E$="RC+E+C+E+C+G+C+  RC+F+C+AC+>C+<C+"
1360 H$="C+4.C+C+4E+4     F+4.F+F+4E4"
1370 B$="F+@6"+A$+"@18&"
1380 "!"
1390 A$="A2G+AG+F+      F+4.E16E@60RC+16D+"
1400 D$="R1             I01O4E4G+4>C+2"
1410 E$="AC+F+C+D+CG+C R4E4G+2"
1420 F$="R1             O4C+4C+4E2"
1430 H$="D+4.D+G+4.G+  C+4<B4A+2"
1440 B$="G+@6"+A$+"@6&":A$=A$+"16"
1450 "!"
1460 A$="E4R4G+F+EC+16~2F+16&  F+2R<B>EF+"
1470 D$="I02<G+1                I01E4G+4A4>E4"
1480 E$="I02E1                  I01C+4E4E4>C+4"
1490 F$="R<A+>C+<A+>C+<A+>E<A+ A4>C+4C+4A4"
1500 H$="F+4.F+F+4.F+           B4.BB4.B"
1510 B$="D+@6"+A$+"@18&"
1520 "!"
1530 A$="G+A16B@60R<B>EF+ G+A16B@60R<B>EF+"
1540 D$="I02<B1           B1"
1550 E$="R<EF+EBEF+E      RD+F+D+BD+F+D+"
1560 F$="R1               R1"
1570 H$=">E4.EE4.E        D4.DD4.D"
1580 B$="F+@6"+A$+"@18&"
1590 "!"
1600 A$="G+A16B.AG+AB>C+& C+<A@120RB16A"
1610 D$="B1               A1"
1620 E$="RC+E+C+E+C+G+C+  <ABB>C+F+2"
1630 H$="C+4.C+C+4E+4     I02F+1"
1640 B$="F+@6"+A$+"@6&":A$=A$+"16"
1650 "!"
1660 A$="_2BA16A@60G+AG+F+ F+4.E16E@60RC+16D+"
1670 D$="I01O5ACD+AG+4F+4  EC+EG+R@6>C+@90"
1680 E$="R2O5D+4D+4        C+RRRR@3G+@93
1690 F$="R2O4B4B4          R2O5E@96"
1700 H$="I01 A2G+2         >C+4<B4A+2"
1710 B$="A@6"+A$+"@6&":A$=A$+"16"
1720 "!"
1730 A$="E4EF+E4D+E"
1740 C$="R1"
1750 D$="O4F+A>C+EG+2"
1760 E$="R2E2"
1770 F$="R2C+2"
1780 G$="R2O4A2"
1790 H$="<F+2B2"
1800 B$="D+@6"+A$+"@18&":A$=A$+"&"
1810 "!"
1820 IF R=1 THEN 2100
1830 ' --<< D >>--
1840 A$="E2.~2R4       R"
1850 C$="R1            R1"
1860 D$="G+A16B@108>E4 <F+G+16A@156"
1870 E$="I02F+1        E1"
1880 F$="I02O3B1       O5C+1"
1890 G$="R1            R1"
1900 H$="I02>E1        B1"
1910 B$="E@6"+A$+"@186":A$=A$+"@192"
1920 "!"
1930 A$="R1   R1"
1940 B$="R1   R1"
1950 F$="R1   C+@191I01"
1960 H$="O4E1 E1"
1970 "!"
1980 D$=        STRING$(2,"O6F+DEC+D<B>C+<A")
1990 E$="I01"+STRING$(2,"O6D<B>C+<ABGAF+")
2000 F$=        STRING$(2,"O4RB4.R>D4.")
2010 H$="I01"+STRING$(2,"O4G2G2")
2020 "!"
2030 D$=STRING$(2,"O6FCD<B>C<ABG")+"E4<F+>ED+2"
2040 E$=STRING$(2,"O6C<ABGAFGE")  +"<A4R4A2"
2050 F$=STRING$(2,"O4RA4.R>C4.")  +"R2<F+2"
2060 H$=STRING$(2,"F2F2")         +"O2B>F+B4<B2"
2070 "!"
2080 R=R+1:GOTO 600
2090 ' --<< E >>--
2100 A$="E2.~2R"
2110 C$="R1"
2120 D$="E2R<B>EF+"
2130 E$="<B2RR>C+4"
2140 F$="<G+2RRA4"
2150 G$="R1"
2160 H$="O2EEEEEEB4"
2170 B$="D+@6"+A$+"@42":A$=A$+"@48"
2180 "!"
2190 A$="R1     R1"
2200 B$="R1     R1"
2210 C$="R1     R1"
2220 E$="E2R2   D+2R2"
2230 F$="B2R2   B2R2"
2240 G$="R1     R1"
```

```
2250 H$="E4.EE2 D+4.D+D+2"
2260 D$=STRING$(2,"G+A16B@60R<B>EF+")
2270 "!"
2280 A$="R1                R2.RB16A"
2290 D$="G+A16B.AG+AB>C+& C+<A@120R4"
2300 E$="E+2.G+4           F+2.R4"
2310 F$="B2.>E+4           C+2.R4"
2320 H$="C+4.C+>C+4E+4     F+4.C+<F+2"
2330 B$="R@6"+A$+"@6&":A$=A$+"@12"
2340 "!"
2350 ' --<< F >>--
2360 A$="BA16A@60G+AG+F+     F+4.E16E@60RC+16D+"
2370 C$="R1                  R2O7R@6C+@90"
2380 D$="O6ACD+AG+4D+4       EC+EG+R@3G+@93."
2390 E$="R1                  R2O6E@96"
2400 F$="O5R@6A@90R@6G+@90   R@6G+@90R@6G+@90"
2410 G$="O5R@3F+@93R@3D+@93 R@3E@93E@3E@93"
2420 H$="O5D+@96C@96         C+@48<B4A+@96"
2430 B$="A@6"+A$+"@6&":A$=A$+"@12"
2440 "!"
2450 A$="E4EF+E4D+.E16& E2.R"
2460 C$="R1             R1"
2470 D$="F+A>C+EE4<B4   <G+A16B@156"
2480 E$="R2O7C+4R4      R2O5EG+4."
2490 F$="R2O6A4R4       R1"
2500 G$="R1             1O2O3B1"
2510 H$="O5F+2<B2       IO2O3E1
2520 B$="D+@6"+A$+"@42":A$=A$+"@48"
2530 "!"
2540 A$="R1            R1"
2550 B$="R1            R1"
2560 D$="O5F+G+16A@156 EC+F+D+E4D+<B16>C+16&"
2570 E$="R2D+F+4.      <G+4G+4>C+4<F+R16A16&"
2580 F$="R1            R2O4G+4R.E16&"
2590 G$="IO2B1         R1"
2600 H$="D+1           IO1C+4<B4A4G+.F+16&"
2610 "!"
2620 A$="T80R2T76R2  T72R4T66R4T60R4T36R4"
2630 D$="C+2R@6<A@90 R<B>EF+G+B> R@4G+@188"
2640 E$="A2R@3E@93   R2.R@2>D+@190"
2650 F$="E2C@96R2.   B@192"
2660 G$="R1          R1"
2670 H$="F+2B2       IO2E1"
2680 "!"
2690 END
```

```
2700 '--<< TONE SET >>--
2710 AD=&HB190
2720 READ A$:IF A$="END" THEN RETURN
2730 POKE AD,VAL("&H"+A$)
2740 AD=AD+1:GOTO 2720
2750 '--<< TONE DATA >>--
2760 '+-----------------------------------------+
2770 '| 01:Piano       02:Piano       03:Vocal   |
2780 '+-----------------------------------------+
2790 DATA FA,00;51,25,71,11,25,3E,4D,00,5F,56,5D.9F,05,00,00,07
2800 DATA 07,04,04,06,94,45,45,45,00,80,80,00,00,00,80,00,00,00
2810 DATA FA,00,51,25,71,11,25,3E,4D,00,5F,56,5D,5F,05,00,00,07
2820 DATA 07,04,04,06,94,45,45,45,00,80,80,00,00,00,80,00,00,00
2830 DATA FC,00,32,72,72,22,1E,0C,22,0A,1C,0D,1C,0D,07,07,07,07
2840 DATA 05,05,05,05,16,16,16,16,00,00,00,00,00,00,80,00,00,00
2850 DATA END
```

リスト4　ジングルベル

```
 10 /*
 20 /*     ジングルベル
 30 /*
 40 m_init():m_alloc(1,1225):m_assign(1,1)
 50 str a(10)[256]
 60 /*
 70 m_tempo(106)
 80 /*
 90 a(0)="@31v14q818 |:256 o4"
100 a(1)="   v14 ccagfc4rc cagfd4r"
110 a(2)="    ddb-age4r<c dc>b-ga4r"
120 a(3)=a(1)
130 a(4)="v15 ddb-ag<cccc dc>b-gf4.r v14"
140 a(5)="    aaa4aaa4 a<c>f.g16a4r4"
150 a(6)="    b-b-b-.b-16b-aaa16a16 aggfg<c4.>"
160 a(7)="v15"+a(5)
170 a(8)="v14 b-b-b-.b-16b-aaa <cc>b-gf4.:|"
180 /*
190 for i=0 to 8:m_trk(1,a(i)):next
200 /*
210 m_play()
```

（善）のゲームミュージックでバビンチョ

みんな元気か。僕は花粉症だ。このコーナーはなんかひさびさのような気がするな。ひと月休んだだけだけど。

で，みんなはZMUSIC.Xをもう手に入れたかな。表紙を描いてくれたのは「響子in CGわ〜るど」の寺尾響子さんだ。響子さんとシュークリームに感謝。

●STREET FIGHTERⅡ IMAGE ALBUM
CD：PCCB-00075　ポニーキャニオン
2,500円(税込)

大人気がなお止まない，現在ヒット街道独走中のアーケードゲーム「STREET FIGHTERⅡ(以下SFⅡ)」。Oh！X編集部にも基板があって，対戦が人気沸騰中。

で，そのSFⅡのイメージアルバムが今回発売となる。収録内容は，基本的にゲーム中のBGMのゴージャスアレンジバージョンといった感じ。少々打ち込み臭い(R8でしょ，WSでしょ，ええと……)けれど，シタールびろろんのインド調あり，鼓ぽこぽんの和風サウンドあり，ハウス・ブラック調ありフュージョンあり……と，なかなか工夫された飽きさせない構成。すでに発売中のオリジナルサウンドアルバムと併せて購入すると，2倍楽しめるかも。

お勧め度　8

●茶々丸ゲーム・ミュージック（ヒューマン・オムニバス）
CD：TKCA-30378　徳間ジャパン 2,300円(税込)

ゲームボーイソフトらしいが，はっきりいって全然知らない。聞いたこともなかった。でも，届いたテープを試しに聞いてみたら思いのほかよかった。

内容はいつものパターン，オリジナルサウンドとアレンジバージョンという構成。アレンジバージョンが秀逸で，1，2トラックのサンバ調の曲調が新鮮で，メロディラインも美しいと感じた。オリジナルサウンドのほうは，ゲームボーイだけあってPSG数声の貧弱なもの。こっちは少し聞きづらかった。

お勧め度　7

読者からのお勧め

埼玉県の醍櫂令さんからのお便り。

●ALL OVER XANADU　CD：BY30-5170
アポロン　3,000円(税込)

「西川さん，いつもいつも馬鹿な原稿を楽しく読ませていただいております。私が前から紹介したかったCDがこれです。日本ファルコムのARPGの傑作「XANADU」と「XANADU SCENARIOⅡ」のBGMのアレンジバージョンが収録されています。バロック調にロックの味付けのなされたなんともいえない曲調に，もう体がシビレちゃいますよ」

善：なるほど。本当にこりゃあいい。これを聞いて初めて知ったんだけど，古代祐三って「XANADU SCENARIOⅡ」の曲も作曲してたんだね。

おや，こんなところにもう1枚葉書がきてるぞ，ナニナニ……？

東京都　門巣太さんからのお便り。

●モンスター・レアー　HCD9006
ハドソンソフト　5,800円(税別)

「こんにちは，はじめにいっておきますが，これは音楽CDソフトではありません。PCエンジンのCD-ROM²のソフトです。先日中古ソフト店を覗いたら安売りしていたので買ってみたら，ゲームはソコソコなのですが曲が妙にいいのです！　トラック12に収録されている1面の曲なんか最高です。ぜひ聞いてみてください。当たり前ですが，曲だけなら普通のCDプレイヤーで聴けますよ。ちなみに全曲アレンジバージョンですので，音的にもグーです」

善：私はファミコンもPCエンジンも持っていないので，この方面はうといのだ。なるほど，これもサンバ調だね。サンバっていうのはこういうメロディが綺麗で覚えやすいものが多いね。

終わりに

ZMUSIC.Xは本屋さんで注文して買ってね，て宣伝ばっかりだな今月は，ってひとりで突っ込んでると古村君みたいだな。

では，また来月。





メロディを生かす伴奏とは？

Taki Yasushi 瀧　康史

今回は簡単なコード進行の仕方を理解したうえで，いよいよアレンジの基本ともいえる伴奏に挑戦してみましょう。ただ，リズムはまだよくわからないと思いますので，今回は簡単な伴奏パターンを取り上げてみました。

夏に出会う約束は

第1回で紹介したPre・Primerの2作目が出ていました。これ，静岡に里帰りしたとき，パルシェ（駅ビル）でいきなり見つけたんですよね。青い化粧箱の中に入って。私自身はパソコン系の雑誌は本誌ぐらいしか毎月読んでないし，ほかの雑誌はめぼしい記事があったときに買うくらいなので，こういうアルバムが出てるなんてことは知らなかったんです。

ひととおり聞いてみてまず思ったことは，前作と違って結構原曲のメロがわかってしまうこと。ま，これはいいとして，前作よりもなおいっそう，グランドピアノの色が濃く出ています（グランドピアノほしいなあ……）。それがとっても澄んでいて，きらきらと星がきらめくイメージが伝わってくるって感じでいいんです。この中で私がいちばん好きな曲は，2曲目ですね。グランドピアノがすごく活躍してくるところからが，ほんっとに好きです。根本的にピアノの音が好きなんですよね，私って。それから，グランドピアノの遊びっていうのかな？　楽しむように気楽に進む。まるで，ストリングスのメロディを無視してるかのようで，しっかり礎をしつつも美味しいところを持っていく，あのグランドピアノのリフが好きだなあ。

私の友人の長谷川君は，このCDはゲームミュージックだということで損をしているといっていました。なるほど，ごく一般的に考えれば，ゲームミュージックなどは音楽ではないといっている人もいますし，そうまではいわなくても，ゲームミュージックだというだけで，敬遠してる人もごく当たり前にいるようです。曲の趣味を人に対していう筋合いはありませんし，権利もありません。逆もそうです（私のような立場にいる場合，逆は必ずしも成り立ちませんが）。ただ，ゲームミュージックに対して，敬遠している人たちに，私の立場からいえる言葉があるとすれば，それはいろんな曲を聞いてみるのもいいかもしれないよ，とまあそれくらいですからね。

ちょっと前置きにしては長くなってしまったかな？　今回は感想というか，個人的な好き嫌いになっちゃいましたけど，Pre・Primer，かなりいいアルバムだと思いますよ。特にウィンダムヒルや，西村由紀江なんかの曲が好きな人には特におすすめです。ぜひ聞いてみてください。

Chord Progression 終止形

さてと。終止形という言葉を聞いて，寒気を感じた人，何人います？　実のところ，私も以前は結構寒気を感じていました。なぜって？　これが結構ややこしいんです。だから以前終止形についてまでは勉強したんだけど，ここで挫折したことがある人がいるんじゃないかな？　って思って。

終止形とは，ちょっと難しめにいうと，楽曲を構成する最小単位のコード進行のことです。ジャズをやってる人がよくいうコードパターンというのも，だいたいはこのことです。

前回で基本的なコードをひととおり押さえたことですし，今回は簡単なコード進行について考えてみましょう。とりあえずトニック，サブドミナント，ドミナントのみで構成するコード進行だけを取り上げてみたいと思います。コード進行の本には，原理だとかそんなことが大変たくさん書いてあります。でも，原理はあとから加えればいいと思うんです。だから私の連載ではまず例をとってみて，それから復習とうんちくという形で理論を加えたいと思っています（え？　もう理論が先行してる？）。

さてご存じトニック，サブドミナント，ドミナントには基本的な性質がありましたよね？ 第2回の復習のコラムに特に注意深く載っていますが一応，

トニック…………どのコードにも進行することが可能
ドミナント………トニックのみに進行することが可能
サブドミナント…トニックとドミナントに進行することが可能

という性格がありました。それぞれ，そんなに難しく考える必要はないわけで，もう身についているんじゃないかなと思います。

こんなところを前知識として，図1のそれぞれの楽譜を見てください。全部で3種類あります。非常に基本的なコード進行と，それに見合ったどっかで聞いたことがあるメロディです。どうですか？楽器のある人はちょっと弾いて見てください。どれもどっかで聞いたようなメロディですよね？誰でも小学校できっと習ったでしょう？

今回使用するコードは基本的には，トニック，ドミナント，サブドミナントの性格を考えるとすぐ出てくる，ごく基本的なものです。

10月号の最後でやったような表記をして，Ⅰ，ⅰ，Ⅴ7，Ⅳ，ⅳと表してみましょう。頭の数字はそのスケールの根音から何度の音かという意味ですから，仮にスケールCならC，Cm，G7，F，Fmとなります。なぜ，素直にコードをそのまま書かないかというと，平行移動して違う調に持っていくことができるからです。

それでは，各々についてちょっとだけ説

明を入れてみましょう。

1．I－V7－I（i－V7－i）

（A）「ちょうちょ」の歌の出だしです。誰でも聞いたことありますよね？ ご覧のとおり，I－V7－I，ここでは，スケールをC（maj）としているので，C－G7－Cとなっています。実のところ，この曲は全体にわたってIとV7しか使われていません。いってしまえば，この2つだけでも名曲は作れるってことです。え？「ちょうちょ」って名曲かって？ う～ん賛否があるにしても，いろんな人がみんな知ってるんだから，名曲なんでしょう。

（B）別に曲でもなんでもありません。ほら，子供のとき学芸会や発表会で「挨拶」って感じでやりましたよね？ 実はI－V7－Iの進行だったんですね。

（C）ロシア民謡の「トロイカ」です。テトリスでありましたよね？ ただ，マイナースケールでも同じようにあるという意味でのサンプルです。

2．I－IV－I（i－iv－i）

「チューリップ」の最後のところです。

3．I－IV－V－I（i－iv－V－i）

曲名は忘れてしまったのですが，確か小学校のとき，リコーダで練習した覚えがあるんですけどね。

＊

というわけで，このトニック，ドミナント，サブドミナントの性格から，基本的なコード進行はメジャースケールから3種類，マイナースケールから3種類できます。メジャーとマイナーは基本的には，曲の中に混ざらないはずですから（転調する場合を除いて）3つしかありません。かといって，別にスリーコードのみで作られた曲が単調になるわけでもありません。いつか話したように，スリーコードでそのスケールのすべての音を包含してしまうからです。スリーコードの名曲とかよくいうでしょう。

伴奏をつける

楽曲の3大要素は何か？ それは，いわずと知れたメロディ(旋律)，リズム，ハーモニーです。いきなり曲を作るまではいかなくても，ここでは非常に簡単な曲をサンプルに伴奏をつけてみましょう。

伴奏をつけるっていうのは私はもっとも基本的なアレンジだと思っています。そういえば，さっきの「チューリップ」のような曲は正しい伴奏なんてあるのでしょうか？ とりあえず伴奏をつけるにあたって，ポイントを自分なりに考えてみました。

1） メロディを殺さない

2） ハーモニーをくずさない

3） 曲に厚みやメリハリを持たせる

というわけで，ほとんどもろに楽曲の3大要素と類似してしまいました。考えてみれば当たり前なんですけどね。リズムについては3番に包含されるでしょう。

では，この3つについて順を追って考えましょう。

1） メロディを殺さない

いつか，メロディは結構いいかげんにできている（に近いことを）といいました。はっきりいって，自分はどちらかというと，「いいコード」を追求するより，「このメロディ！」「この旋律！」を重視するほうなので，メロディを生かすか殺すかは死活問題です。

確かにメロディはいいかげんなものかもしれません。伴奏次第で華やかにも，悲しくもなったりします。では，悲しい（作ったつもりの）メロディに楽しい伴奏をつけたらメロディを殺したことになるか？ これは問題です。私としては，逆に逆手を取ってメロディを生かしたことになると思います。それが変に聞こえなく，無理なく聞こえれば，ですが。

どちらにしても，音楽は「イメージ」だと思いますからね（しかし，昔は音の数学だと考えられていました）。

2） ハーモニーをくずさない

曲のハーモニーをくずす簡単な方法に，まるっきり合わない音を鳴らす。ということがいえます。逆にいえばはずれない音をださなければ，ハーモニーをくずさないことになるのでは……(大嘘！)。とまあ，難しいですから簡単に考えてみましょう。

それでは，はずれない音というのはなにか？ ちょっと簡単なメロディラインから考えてみましょう。

生贄は，とってもメロディラインの簡単な「ちょうちょ」です。スケールをC(Maj)にするのなら，最初の1小節目のコードはCです。まだ，メロディからコードは導き出せない人がほとんどだと思うので，ここではメロディとコードがわかっているもの

図1　基本的コード進行

(1) A　C　G7　C

B　C　G7　C

C　Dm　A7　Dm　A7

(2)　C　F　C

(3)　C　F　G7　C

と考えます。

コードがわかってるなら，はずれない音を出すのは簡単です。まずCのコードの構成音は，C，E，Gですね。とすると，当然のごとくはずれない音はC，E，G。

図2を見てください。当然，±nオクターブ上の音や下の音もはずれませんよね？

長い音，すなわち白玉（全音符，2分音符）や，アルペジオによってハーモニーを出そうとするときなどは，特にこれ以外は使わないほうが無難です。

次にはずれないコードといったら，スケール上の音でしょう。でもこれは問題です。通常使う場合，一定の長さ分の音を出すときは，いくら同じスケール上といっても，コードの構成音は変に聞こえてしまいます。だから，はずれない音というより，はずれ難い音とイメージしておきましょう。

3) 曲に厚みやメリハリ持たせる

さて，上の2つがいわば「やってはならないこと」「約束事」ならば，この3項目めはいわば「目的」「その人個人のオリジナリティ」になるでしょう。

音楽を作るうえで厚みを持たせるというのも重要な要素でしょう。「ちょうちょ」のような曲にドビュッシーのような厚み……う～ん。怪しい，怪しすぎる……。

当然のごとく分厚い音だけ鳴らしても，音楽は分厚くはなりません。どうすれば厚くなるか。答えは簡単，いろんな成分の周波数を含む，すなわちいろんな音を入れる。それも，1)，2)の規則を守らなくてはならないと（なんかSX-WINDOWのプログラムに似てるなあ）。復習でやったようにそれは当然トライアドよりクォードのほうが厚みが出ます。その辺は次のメリハリの部分も含めて「技」でしょう。

メリハリというのはいわば変化です。曲に変化を持たせるのは一概に伴奏だけでやるものではありません。ただ，同じようなメロディでも伴奏によって違った雰囲気の曲を作ることは可能です。適当ないい例がないので作るか探しておくことにしましょう。

図2 コードの構成音

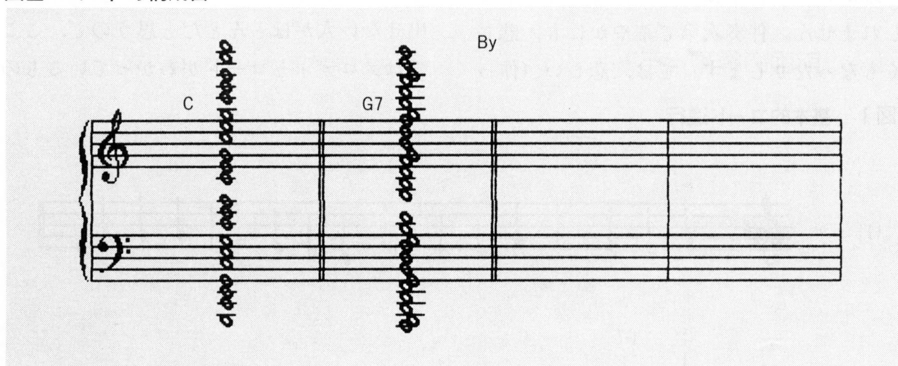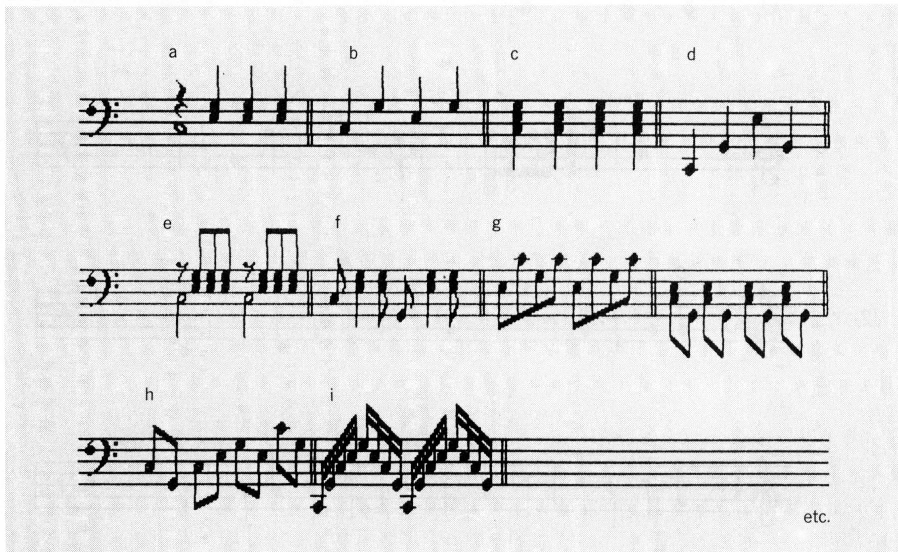


図3 左手の伴奏の例



リズムも重要な要素ですが，今回ははしょることにします。リズムについては1回かけていつか説明したいと思います。

伴奏をする楽器というのもいろいろありますが，ここはいちばんポピュラーなピアノを例にとって考えてみたいと思います。

たとえば，ピアノ曲は基本的に「右メロディ，左伴奏」がよくあるパターンです。これがショパンやベートーベン，モーツアルト，ラフマニノフと，クラシックピアノ系の曲になるとどうにもならなくなるわけですが，ここは強行！ 右手はメロディ，左手は伴奏ということにします。

そこで，左手の伴奏パターンを考えてみたり，いろんな曲を弾いた記憶からピックアップしてみました。4ビート（4拍子）で話を進めましょう。図3を見てください。ご覧のとおりetc.なんですよ。変じゃなければなんでもはまります。すべてコードCを基準として作られていますから，Cの構成音すなわち，C，E，G以外の音は使われていません。逆に使った伴奏パターンもできるのですが，ここではややこしくなるのでやりません。匂わせておくと，sus4なんか……ま，コード構成音以外は慣れるまで使わないことをすすめます。

aがいちばんの基本パターンですね。スローテンポでもハイテンポでもミドルテンポでもなんでも合いますが，ちょっと単調です。たいていの曲はこれで合うんですけどね。合わない曲もあります。これが8ビートっぽくなると，fみたくなります。

bはミドルテンポが合うでしょう。「ちょうちょ」はこれがいちばん合うんじゃないでしょうか？ 私のセンスって貧弱かなぁ？ cは比較的ミドルテンポ，Andante（アンダンテ＝ほどよい速さで）ぐらいのテンポでしっかりリズムを刻みたいときにいいでしょう。

eあたりは，私が好きな伴奏です。Grave（グラーヴェ＝重くゆっくりと）ぐらいのテンポで，しっかり重く弾くことによって，分散和音でありながらなかなか安定させることができます。

gなんかは結構ひょうきんなリズムです。たとえば，Ysのお店の曲のリズムはこんな感じじゃなかったかな？

hは，分散和音の典型です。アルペジオ自身分散和音なのですが，モーツアルトに

よく見られる型でしょう。ちなみにハイテンポでこれを弾くと凄まじい曲にすることができます。

iは，4ビート系の曲では基本中の基本です。いろんな曲でこのパターンを使うことができます。

*

そんなわけで，伴奏パターンをいろいろ紹介してみました。だいたい決まったパターンというのがあるので，あとははずれない音，すなわち，そのコードに乗った音ならいいわけです。そう考えると，コードがすでにわかっている曲なら，音を付け加えることがもうできますよね？ あとは，それなりにあった伴奏パターンを当てはめればいいということになります。

おわりに

子供の頃に，音楽の先生の即席アレンジの「技」をみて「凄いぃぃぃ」って思ったことがあります。教科書には，歌の楽譜しか書いていないのに，ピアノ（もしくはエレクトーンでも。昔はオルガンってのもありましたよね）でそれを見ながら，すらすらと，伴奏を作ってしまう先生って凄いなって。実は先生用の音楽の教科書には伴奏の楽譜も書いてあったのですが。ま，そんなことはどうでもいいとして，その曲中のコードがわかり指さえ動けば，たとえ先生用の音楽の教科書がなくっても，似たようなことができるようになります。

ただ，ここまでやってきたことをすべて身につけていたとしても，メロディを聞きながら，コードを押さえることはまだできないでしょう。

次回は「メロディラインからコードを導く方法」と，もし余裕があればトニック，ドミナント，サブドミナント以外のコードの使い方をやってみたいと思っています。これまでの連載のなかで，「ここがわかんないよ」とか「いまいちつかめないよ」なんてのがあったら，どんどん質問してください。

あ，これまでやったこと以外でも結構。「こういうことをやってみたんだけど」とか，こんな曲をアレンジしたいんだけどかから，曲を作りたいんだけど，いまいちこのことの意味ががわかんなくてうまくいかない，みたいな，意欲的な質問もバリバリしてください。

今日やった伴奏つけオンリーのようなアレンジなら覚えるのは比較的簡単です。でも，私個人で思うのは，「余分なメロディをつける」ことと「オブリガッドを入れる」ことに面白さがあると思います。そうなると，やっぱり作曲の知識まで必要になっちゃうんですよね。でも，身につき始めたときの面白さはそう簡単に口ではあらわせません。

「楽器がないからアレンジできないよ」そういってしまえばそれまでです。確かに，パソコンの近くに鍵盤があって，楽譜とペンがある環境より，身につけるのは難しいと思います。でも，内蔵音源だけでもある程度はできると思いますよ。

ひとりでも音楽をすることが好きな人が増えることを祈って……次回またお会いしましょう。

前回の復習とうんちく

とりあえず前回やったことを整理してみましょう。

まずトニック，ドミナント，サブドミナントの三種類。曲を構成するもっとも基本的なコードとなります。とくにトニックは曲中で，どんなコード進行にも進め，なおかつそのスケールを代表するものです。そしてドミナント。スケールの根音から数えて5度の音を基音にしたコードです。これを説明する前にちょっと余談をしましょう。

編集部でいろいろ話しあったんですけど，どうもそのスケールの根音をそのスケールでの「ド」という教え方があるらしい。え？ 余計わからない？ 要するに，G(maj)のスケールがありますよね？ GABCDEF#Gって羅列です。このGにあたる音を，「根音」というのはもうおわかりかと思いますが，中学（もしくは高校？ ひょっとしたら小学校？）でト長調の「ド」という習い方をしているそうなんです。ドレミファソラシドって相対音階？ でもそれはわかりやすいように，誰かが考え出したような気がするし，馴れないことをやるとそれでなくても多いぼろがぼろぼろ出る可能性が高い。それに絶対音階と区別が面倒ですから，私は○長調もしくは○短調の「ド」という使い方はしないで，「根音(ルート音)」といういい方をします。

さて話を元に戻しましょう。最初に述べた「スケールの根音から数えて5度の音を基音にしたコード」というのは，G(maj)にたとえると，スケールの根音＝G，から5度の音＝Dを基準にしたコード＝D(maj)ということになります。前回いったとおり，このドミナントコードは7thを含むことによって，トニックに進んだとき（これをドミナントモーションという）より安定感を得ることができます。7度の音は不協和音でしたよね？ 不安定から完全安定に進むと曲想的によりよい安定感がえられると。そんな訳ですね。

では次に，なぜトニックに強く進行したがるか，ドミナントモーションはなぜ安定した進行なのか？ を復習しましょう。

例によって例のごとく，C(Maj)スケールを基準に考えてみましょう。このスケール中のドミナントはもう知ってのとおりG7です。このコードの構成音はGBDFですよね？ GBDをとると当たり前，これはもう長3和音です。では，このうち根音（この場合基音と同義語です）をなくして………というかとっぱらってしまって，BDFにしてしまうとなんと減3和音（ディミニッシュトライアドコード）です。このBを根音とする減3和音BDFで，減5度音程にあたる音（F）と根音（B）の2音は引き合って，協和音程である3度音程の和音（この場合）トニックCに解決しようとします。すなわちBがC，FがEになる訳です。ちょっと見ると，お互いに引き合ってるように見えますよね？ これがドミナントモーションというんです。

次。5つのトライアドコードと7つのクォードコードについて説明しました。トライアドコードは3音で構成されるコード，クォードコードは4音で構成されるコードです。トライアドコードの種類は，基音をCとしますとC，Cm，Caug，Cdim，Csus4と5種類あります。これからなにかひとつの音を加えたのがクォードコードです。大抵のトライアドコードはクォードに直せるので，何かのアレンジの際，1音加えて比較的簡単に厚みを出せることができるでしょう。それぞれを説明しますと基音をCとした場合，長6度の音が付加されたのがC6。主にトニックでCとCmの代わりに使われます。それから長7度の音が付加されたCMaj7，CmMaj7。主にEmの代わりに使われます。根音CMaj7のCを省略したらEmですからね。

次，短7度の音を付加したC7，Cm7があります。これがドミナントに使われるんでしたね。また，これらの系統とはちょっと違ったCdim（ディミニッシュと読む）があります。Cdimは，それぞれ短3度ずつの音を全部で4つ重ねたコードで，結構怪しい響きがあるわけで，構成音は基音がCの場合，C，Eb，F#，Aの4つです。ダイアトニックトライアドコードのときに説明したディミニッシュトライアドコードから，7度の音を加えて作られています。前回，不思議に思った方もいたと思うんですけど，Cに比べてAは長6度の音じゃないか？ といわれればそうですよね？ でもAは減7度ともいえるのです。度数については，あとで詳しくお話しするつもりです。

さて，これだけの知識を身につけたうえで本文に移りましょうか。

ANOTHER CG WORLD IN KYOTO



なんかサンタにしちゃ人相わるいなー





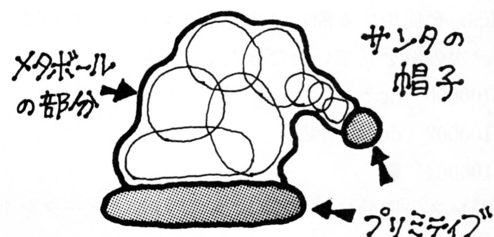


そーして，サンタのおじーさんは，
プレゼントを買いに 出かけてゆきましたとさ

BROROOOO....

今回のCGデータ

総物体数　374
うちメタボール数　20
雪の結晶はポリゴン
光源　6
1280×1024ピクセル
1670万色フルカラーを
4×5ポジで出力
使用ソフトは，C-TRACE，
サイクロン，Z'sTRIPHONY
メタボールは，メタエディタ
がないので，数値を変えて
何回も計算しなおします。
これが，けっこうめんどくさい。





吾輩はX68000である
[第8回]

愛のIOCSコール

Izumi Daisuke　泉　大介

グラフィック画面だからといって
文字が描けないわけではない
今回は大きな愛をお目にかけたい



吾輩がIOCSコールとして携えたグラフィック描画機能をお届けしている。前回はIOCSコールBB_Hの円を描く機能まで紹介したが，試していただけただろうか。既に表示されている色を考慮に入れた描画を行えないという制限はあるものの，なかなか便利に使えることがおわかりいただけたことと思う。今回はさらにグラフィック画面に文字を表示するIOCSコールなどを紹介しながら，吾輩の能力のさらなる深遠へ諸兄を誘いたいと思う。

文字を表示する

ここでいう文字表示はテキストVRAMではなく，グラフィックVRAMへの文字表示である。吾輩X68000がディスプレイに表示する文字の話をしたときに，X-BASICのSYMBOL関数を使って文字を表示したのを覚えていらっしゃるだろうか。文字を拡大表示し，吾輩がドットの集合で文字を表現していることを確認していただいた。これは，その機能のIOCSコール版である。

図1　グラフィック画面に文字を表示する

```
-an 100000
00100000    move.w  #16.d1          ← 768×512ドットモード
00100004    moveq   #$10.d0
00100006    trap    #15
00100008    moveq   #$90.d0         ← グラフィックON
offset overs
0010000A    trap    #15
0010000C    movea.l #$100020.a1     ← 文字情報のアドレスをセット
00100012    moveq   #$bd.d0         ← 表示
offset overs
00100014    trap    #15
00100016    _exit                   ← 終了
00100018    .
-an 100020                          ← 文字情報をセット
00100020    dc.w    0,200           ← 表示する座標
00100024    dc.l    $100030         ← 文字の入っているアドレス
00100028    dc.b    5,5             ← 倍率・縦横5倍
0010002A    dc.w    15              ← 色は白
0010002C    dc.b    2,0             ← 24ドットフォント 回転なし
0010002E    .                       ← 終了
-an 100030                          ← 文字をセット
00100030    dc.b    '愛',0
00100034
```

このIOCSコールBD_Hは表示する文字の情報を収めたアドレスを，A1レジスタにセットして使用するようになっている。図1をご覧いただきたい。これはグラフィック画面に文字を表示するプログラムである。文字情報をセットしているのはアドレス100020_H以降で，情報は以下の順にセットしてある。

1)　文字を表示する座標(X，Yの順にワード単位)
2)　表示する文字列を格納したアドレス
3)　縦横の倍率(縦横の順にバイト単位)
4)　文字の色(ワード)
5)　使用するフォント
　0：12×12ドットフォント
　1：16×16ドットフォント
　2：24×24ドットフォント
6)　文字を表示する角度
　0：回転なし
　1：90°
　2：180°
　3：270°

図1ではデータセットに「dc」という命令を使っている。これはMC68000の命令ではなくアセンブラの命令で，データを定義するのに使用する。「.b」をつければバイト単位に，「.w」をつければワード単位に，「.l」をつければロングワード単位にデータをメモリに張り付けてくれる。もちろん，デバッガのMEコマンドを使ってデータをセットしても構わないが，100030_Hのようにして表示文字列を格納できるのは「dc」のメリットといえるだろう。

「dc.b」を使用する際には注意していただきたいことがある。次の例を見ていただきたい。

```
100000  dc.b    1
100002  dc.b    2
100004  ■
```

アドレスの増分に注意されたい。バイトデータを1つしかセットしていないのにアドレスは2バイト分大きくなっている。これは，MC68000の命令は必ず偶数アドレ



スに置かなければならないというルール故，A/ANコマンドはアドレスが偶数になるようたえず調整しているためである。このため，図1の文字情報の倍率やフォント選択の部分を2行に分けて書くと，正しいデータがセットされないことになる。「dc.b」で奇数個のデータをセットした場合には必ずこのような問題がついてまわる。AS.Xなどのアセンブラではこのようなことは起きないが，反面，

```
    dc.b    '吾輩はX68000である',0
    dc.b    '名前はまだない',0
```

のように文字列をセットする場合に，2つめの「名前はまだない」を偶数アドレスから始めるためには，「.even」を，

```
    dc.b    '吾輩はX68000である',0
    .even
    dc.b    '名前はまだない',0
```

のように挿入しなければならない。これはアドレスを偶数バイトから始めることを指示するアセンブラの命令である。

図1のプログラムのほうはまことに他愛ない。例によってIOCSコール10_Hで画面モードを設定し，IOCSコール90_Hでグラフィック画面を消去・グラフィックONにする。続いてIOCSコールBD_Hで文字を表示すればOKである。

プログラムの最後，100016_Hは「EXIT」というDOSコールで終わっている。これは拡張子が「.X」の実行ファイルを終了するためのDOSコールである。デバッガを使ってメモリに直接プログラムを作成している場合には本来使わないDOSコールなのだが，ブレイクポイントを設定するのが面倒だという方のために入れてある。このDOSコールに出会うと，デバッガはそこでプログラムの実行を中止する。まぁ，プログラム実行の最後を示すものと考えられたい。

プログラムの実行は，

```
    g=100000
```

である。大きな愛が現れただろうか。こうして見ると，24ドットフォントも結構粗いフォントであることがよくわかる。ベクタフォントやアウトラインフォントがもてはやされるのも道理といえよう。

文字の粗さ比べ

文字をグラフィック画面に拡大表示するIOCSコールBD_Hを紹介したので，ついでに文字の粗さ比べをしてみたい。普段の文字表示に使用している16ドットフォントのほかに，吾輩はいま紹介した24ドットフォント，そしてビジュアルシェルやSX-WINDOWでおなじみの12ドットフォントの3種類のフォントを標準で表示することができる。24ドットフォントを4倍したものと，16ドットフォントを6倍したものと，12ドットフォントを8倍したものは同じ大きさで表示される。これを比べてみよう。プログラムは図2のようになる。

図1と同じプログラムで文字を表示したら，X座標，倍率，フォント設定を変えて次の文字を表示するという作業を繰り返し，横に3つの解像度の異なる文字を表示している。倍率はバイトデータ2つで指定するが，ここではワードデータを一気にセットすることで簡略化してある。写真でもおわかりいただけるかと思うが，16ドットフォントではかなり貧相になり，12ドットフォントに至ってはとても文字とは思えないひどさである。これでも，画面に実際に12×12ドットで表示され，前後に適当な文書があれば人間は「愛」と読んでしまうのだから恐



図2　文字比べ

```
-an 100000
00100000        move.w  #16.d1
00100004        moveq   #$10.d0
00100006        trap    #15
00100008        moveq   #$90.d0
offset overs
0010000A        trap    #15
0010000C        movea.l #$100060.a1
00100012        moveq   #$bd.d0
offset overs
00100014        trap    #15              ← ここまでは図1と同じ
00100016        move.w  #100.$100060     ← X座標を100にする
0010001E        move.w  #$0606.$100068   ← 倍率は縦横6倍
00100026        move.b  #1.$10006c       ← 16ドットフォント
0010002E        moveq   #$bd.d0
offset overs
00100030        trap    #15
00100032        move.w  #200.$100060     ← X座標を200に
0010003A        move.w  #$0808.$100068   ← 縦横8倍
00100042        move.b  #0.$10006c       ← 12ドットフォント
0010004A        moveq   #$bd.d0
offset overs
0010004C        trap    #15
0010004E        _exit
00100050        .
-an 100060
00100060        dc.w    0.200
00100064        dc.l    $100070
00100068        dc.b    4.4              ← 4倍に変更する
0010006A        dc.w    15
0010006C        dc.b    2.0
0010006E        .
-an 100070
00100070        dc.b    '愛'.0
00100074
```

図3　ファンクションキーモード

モード	ファンクションキー	画面行数
0	表示	31行
1	シフトキー表示	31行
2	非表示	31行
3	非表示	32行

図4　画面モード

モード	解像度	グラフィック
0	768×512	なし
1	768×512	16色
2	512×512	なし
3	512×512	16色
4	512×512	256色
5	512×512	65536色

れ入る。

諸兄の中には日本語入力FEPにFIXERを使用している方がいらっしゃると思う。吾輩が携えたASK68Kと比較すると変換はかなり賢く，さすがは単品の商品だけのことはあると，うちの御仁も大層お気に入りである。先頃，パソコン通信を通じてバージョンアップ用の差分ファイルが配布され，CTRL＋Hなどのコントロールファンクションも実行可能となった(これまではコントロールファンクションが入力されると即座に確定されてしまっていた)。ところが，FIXERを使っていて図1や図2のプログラムを実行すると，少々困った問題が発生する。FIXERはSHIFTキーを押すとファンクションキーの表示を切り替えるのだが，IOCSコール10_Hを実行すると画面が32行モードになるため，画面最下行にコマンドを入力する場合には(しばらくコマンドを入力し続けると必ずこうなる)，入力途中の文字がファンクションキーの表示によって隠されてしまうという事態が起きてしまうのである。また，ASK68Kを使用している場合でも，ファンクションキーの表示が画面から消えてしまい不自由だと感じていらっしゃるかもしれない。

このような事態の発生を防止するには，プログラムの先頭で画面モードを32行モードにし，実行が終了するときに元の31行モードに戻してやればいい。これには「CONCTRL」というDOSコールを使用する。このDOSコールは，画面を消去したり，カーソル位置を移動したり，スクロール範囲を設定するなど，画面(console：コンソール)のさまざまな制御を行うためのもので，ファンクションキーの表示・非表示のコントロールも行うことができる。使い方は，

```
move.w  #ファンクションキーモード,−(sp)
move.w  #14,−(sp)
_conctrl
addq.l  #4,sp
```



である。以前説明したように，DOSコールはスタックにパラメータをセットして利用するようになっている。ファンクションキーモードは図3のようになる。

したがってプログラムを変更するには，

```
move.w  #3,−(sp)   ← 32行モード
move.w  #14,−(sp)
_conctrl
addq.l  #4,sp
```

という4行を先頭に付け加え，

```
move.w  #0,−(sp)   ← 31行モード
move.w  #14,−(sp)
_conctrl
addq.l  #4,sp
```

という4行を_exitの前に付け加えればいい。

なぜこのような面倒な手順が必要なのだろうかといぶかしむ声が聞こえてきそうである。そのわけは吾輩がDOSコールとIOCSコールの2つのサービスをもっていることに関係がある。これらはただやみくもに用意され，適当に2つに分けられたわけではない。ファイルを管理したり，画面を管理したりといった，より高機能なものはDOSコール，ディスクを操作したり，ディスプレイを操作したりといったよりハードウェアに近いレベルのものはIOCSコールとしてまとめられているのである。あるいは，Human68kが必要とする機能はDOSコールに，それを実現するのに必要な下請けの機能はIOCSコールにまとめられていると考えてもいい。

ファンクションキーはHuman68kが管理している。ファンクションキーに文字列をセットしたり，ファンクションキーの表示・非表示を設定するのはすべてDOSコールで行われる。ところが，これまで画面モードの設定にはIOCSコール10_Hを利用してきた。異なるレベルの機能を使ったため，DOSコール内部での辻褄が合わなくなってしまったというのが結論である。つまり，IOCSコール10_Hによって画面からファンクションキーは消去されているのに，DOSコールでファンクションキーを消去していないため，ファンクションキーがまだ表示されているものと判断されてしまうのである。

上のプログラム追加は，この辻褄合わせをプログラムで行うためのものである。また，DOSコール_conctrlも画面モード設定機能を持っているので，こちらを使うことにしてもいい。これは，

```
move.w  #画面モード,−(sp)
move.w  #16,−(sp)   ← 16は画面モード設定
_conctrl
addq.l  #4,sp
```

として使用する。設定できる画面モードは図4のようになる。いずれも31kHzモードしか設定できず，また設定できる解像度も限られているが，DOSコールだけあってファンクションキーの表示にちゃんと対応している。特

殊な解像度を求めないなら，これを利用するのが簡単でいいだろう。

文字を動かす

吾輩のグラフィック機能の概要を紹介したときに，前景の飛行機と背景を異なるプレーンに表示し，背景をスクロールさせることであたかも飛んでいるような表現が可能になるといったのを覚えていらっしゃるだろうか。背景をスクロールさせるゲームが最初に登場したときには，かなりのインパクトをゲーマーたちに与えたものだが，いまではゲームでごく普通に用いられるテクニックとなっている。そのため，ごく簡単なテクニックであるかのように誤解されているような気がする。実際に自分の手でグラフィックをスクロールさせるためには，かなり面倒な作業が必要となる。これはいずれ改めて触れることにし，ここでは吾輩のハードウェアの力を利用したスクロールについて触れておこう。

吾輩のグラフィック画面は512×512，あるいは1024×1024のいずれかである。これはグラフィック実画面と呼ばれている。グラフィックは，この実画面を表示画面と呼ばれる窓から覗くような形で画面に表示される。表示画面のサイズは768×512，512×512，256×256のいずれかである。

最初IOCSコール10_H，90_Hでグラフィック表示状態にしたときには，表示画面と実画面の左上隅の座標は一致している。768×512ドットの表示画面を選べば，図5-1)のように(0,0)－(767,511)の範囲に書き込んだデータがそのまま画面に表示される。そして図5-2)のように表示画面の左上隅の座標を動かせば，実画面左端のデータは表示されなくなり，代わって右のほうに書き込まれていたデータが表示されるようになるのである。これは吾輩の視点から眺めた様子である。固定された実画面があり，その上を表示画面がさまよっているような印象で吾輩はこれをとらえている。この様子を諸兄の視点から眺めると，固定された表示画面があり，その後ろで実画面が動いているような印象を受けることであろう。ディスプレイが机の上で固定されたままだからである。つまり図5-2)では，諸兄は絵が左へ動いたような印象をお持ちになるはずである。これを利用すれば，比較的簡単にグラフィックのスクロールを実現することが可能となる。

IOCSコール$B3_H$は，表示画面の左上隅の座標を設定する。これは，

1) D1.b：ページ指定
   1：ページ0
   2：ページ1
   4：ページ2
   8：ページ3
2) D2.w：X座標
3) D3.w：Y座標

とデータを設定して使用するようになっている。ページ指定は，設定した数値を2進数で考えたとき，第0桁目が1ならページ0を，第1桁目が1ならページ1を……，と決められているので，10進数では上のような数値になる。

では，実験プログラムを見ていただこう。図6である。

図5　ハードウェアスクロールの仕組み

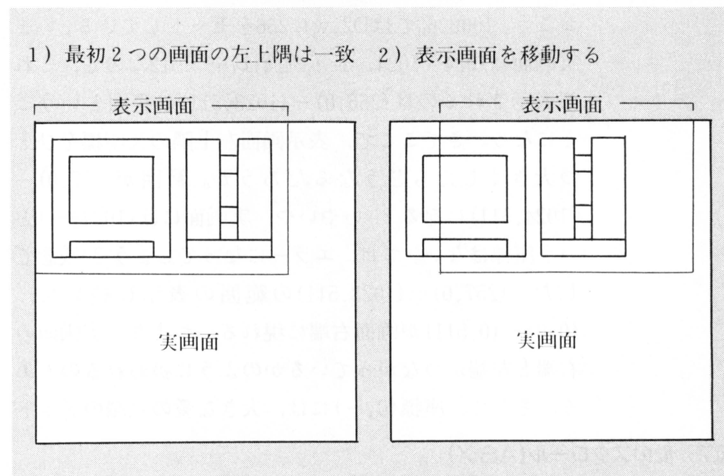


図6　愛のスクロール

```
-an 100000
00100000        move.w  #3.-(sp)
00100004        move.w  #14.-(sp)
00100008        _conctrl
0010000A        addq.l  #4.sp           ← ファンクションキー消去
0010000C        move.w  #16.d1
00100010        moveq   #$10.d0
00100012        trap    #15
00100014        moveq   #$90.d0
offset overs
00100016        trap    #15             ← ここまでいつもの手順
00100018        movea.l #$100060.a1
0010001E        moveq   #$bd.d0
offset overs
00100020        trap    #15             ← 愛の表示
00100022        move.b  #1.d1           ← ページ0をスクロール
00100026        move.w  #256.d2         ← 左上隅のX座標を256に
0010002A        move.w  #0.d3           ← Y座標は0のまま
0010002E        moveq   #$b3.d0         ← スクロール
offset overs
00100030        trap    #15
00100032        move.w  #3000.d0        ← 3000回のループで
00100036        dbra    d0.$100036      ← 時間稼ぎ
0010003A        addq.l  #1.d2           ← X座標を1つ大きくして
0010003C        cmpi.l  #1024.d2        ← 1024と比較
00100040        bne     $10002e         ← 違えば再びスクロール
00100044        move.w  #0.-(sp)
00100048        move.w  #14.-(sp)
0010004C        _conctrl
0010004E        addq.l  #4.sp           ← ファンクションキー表示
00100050        _exit                   ← 終了
00100052        .
-an 100060
00100060        dc.w    0.200
00100064        dc.l    $100070
00100068        dc.b    5.5
0010006A        dc.w    15
0010006C        dc.b    2.0
0010006E        .
-an 100070
00100070        dc.b    '愛'.0
00100074        .
```

ここでは先ほど説明したファンクションキー消去・表示のDOSコールをプログラムに組み込んである。例によってグラフィックをONにしたら，100020_Hまでで画面に愛を表示する。続く100022_Hからがスクロールを行うところで，D1，D2，D3にそれぞれデータをセットしIOCSコールを行っている。

ここで，D2.wにセットするデータについて補足しておこう。100026_HではD2.wに256をセットしている。いま実画面は1024×1024，表示画面は768×512なので，これで表示されるのは(256,0)－(1023,511)の範囲ということになる。さてここで，表示画面左上隅のX座標をひとつ大きくしたらどうなるだろうか。範囲が(257,0)－(1024,511)になる？ いやいや，実画面には(1024,～)という座標はない。では，エラーになってしまう？ 残念でした。(257,0)－(1023,511)の範囲の表示に続いて，(0,0)－(0,511)が画面右端に現れる，つまり，実画面の右端と左端がつながっているかのように扱われるのである。そして，座標(0,～)には，大きな愛の左端のドットがセットされているため，これが画面右端に表示されることになる。以後D2.wを大きくするにしたがって，愛は右から次第にその姿を現すのである。

この「愛が右から次第にその姿を現す」部分を実行しているのが，アドレス$10003A_H$からのプログラムである。ここではD2.w(表示画面左上隅のX座標)をひとつ大きくし，1024になったかどうかをチェックして，まだなら$10002E_H$に分岐してスクロールさせるというループを形成している。その前に，dbraを使った3000回の空ループが入っているのが確認できるだろう。うちの御仁が最初にこのプログラムを作ったときには，この空ループは入っていなかった。結果，愛はあっという間に通り過ぎてしまい，御仁は愛を確認することができなかったのである。いくら吾輩が高速に動作するといっても，3000回の空ループを実行するにはなにがしかの時間がかかる。ここで若干の時間稼ぎをして通り過ぎる愛を確かめようというのが御仁の作戦である。

この程度の大きさのプログラムになると，そしてとりわけBcc, DBRAなどの命令が登場すると，デバッガでプログラムを作成するのはなかなかに困難になる。アセンブラを利用してプログラムを作りたいという諸兄のために，図7に同じプログラムのAS.X版を用意しておいた。AS.X用のプログラムとデバッガ用のプログラムの違いは，

1) アドレスを直接指定することはできないので「ラベル」を利用する
2) _exit，_conctrlなどが定義されていないので，プログラムの先頭で定義しておく

の2点である。

図7 愛のスクロール(AS.X)

```
_exit           equ     $ff00
_conctrl        equ     $ff23

        move.w  #3.-(sp)        * ファンクションキー消去
        move.w  #14.-(sp)
        dc.w    _conctrl
        addq.l  #4.sp

        move.w  #16.d1          * いつもの手順
        moveq   #$10.d0
        trap    #15
        moveq   #$90.d0
        trap    #15

        movea.l #strdata.a1     * 愛の表示
        moveq   #$bd.d0
        trap    #15

        move.b  #1.d1           * ページの設定
        move.w  #256.d2         * X座標
        move.w  #0.d3           * Y座標
scroll:
        moveq   #$b3.d0         * スクロール
        trap    #15
        move.w  #3000.d0        * 時間つぶしの空ループ
loop:   dbra    d0.loop
        addq.l  #1.d2           * X座標増加
        cmpi.w  #1024.d2        * 1024と比較
        bne     scroll          * 違えばscrollへ戻る

        move.w  #0.-(sp)
        move.w  #14.-(sp)
        dc.w    _conctrl
        addq.l  #4.sp
        dc.w    _exit

strdata:
        dc.w    0.200
        dc.l    str
        dc.b    5.5
        dc.w    15
        dc.b    2.0

str
        dc.b    '愛'.0
```

愛が止まらない

IOCSコール$B3_H$で表示画面の左上隅座標として指定するデータは，実画面の大きさを超えることはできない。図6のアドレス$10003C_H$で1024と比較してループを打ち切っているのはこのためである。ここでループを打ち切らず，D2.wを0にして再びアドレス$10002E_H$に分岐するようにすれば，えんえんと愛がディスプレイを回り続けるプログラムとすることができる。具体的には，

```
move.w  #0,d2
bra     $10002e
```

というプログラムをアドレス100044_Hから書き込めばいい。止めるにはCRTL＋OPT.1＋DELでリセットをかけるか，INTERUPTボタンを押していただきたい。

今回は1ページだけのスクロールだったので，スクロールの威力をそれほど実感していただけなかったかもしれないが，原理は理解していただけたと思う。次回はスクロール応用編と題して，悪戯な作品をご覧に入れる予定である。

●特集

音・そして音楽とコンピュータ

コンピュータミュージック，クラシック，ミュージックコンクレート。いかなる呼び名であれ，「それ」は「音楽」に属し，「音」というものから，始まっている。
そして，あらゆる行程を経て最終的に音に還元される。
いま，我々はこれまでの資産を守りつつ，これからの音楽環境をまったく新たに作り直そうとしている。その第１歩を「音」そのものから始めるのは，自然なことではないだろうか。
音自体はなにも特別なものではない。いつしか，どこにでも転がっている「音」が特別な意味を持ち始める。音楽の起源は「感性」だ。かといって「理論」も対立はしない。
音楽が好きだ。
というほど積極的でなくてもいい。
特に好きというわけではないが，なにか曲が流れているのも，悪く，ない。
そんな自然体の音楽環境を作るためにこそ最高の技術が注がれるべきなのだ。「コンピュータミュージック」が忘れ去られるくらい当たり前の概念となる世界のために。

CONTENTS

音とはなにか……………………………………中野　修一
冬の夜長のスペクトル解析 ……………………石上　達也
FM音源の音色を創る ……………………………丹　明彦
Z-MUSIC公式ガイドブック …………………西川　善司
ZMUSIC LIVE SINDO ON STAGE …………進藤　慶到
MIDIをめぐる環境 ………………………………紀尾井　誠
MIDI出力方法論 …………………………………牛島　健雄

音と音源を探る

音とはなにか

Nakano Shuichi 中野　修一

音楽という現象をよく分析していくと最終的に「音」そのものに突き当たります。ここでは音というものを考え，これまでのシンセサイザなどでは，どのようにして音を作っているかを見てみましょう。

娯楽であり，芸術である「音楽」というものを成立させているもの，それを突き詰めてみれば，「音」というものに帰着します。

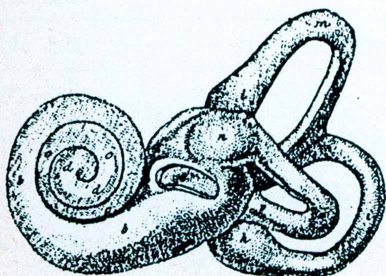
音とはいったいなんでしょうか？ 空気の振動だということは誰でもご存じでしょう。「音波」といわれるように空気中を波として伝わります。まあ，波とはいっても，よく波形表示で出てくるような横波ではなく空気の疎密波，ごく微小な気圧変動として耳に到達し，鼓膜を振動させます。実際に音を感じているのはその先のかたつむり管です(図1)。これは漏斗型の管が丸まった構造をしており，周波数に応じて特定の部位が共鳴して聴神経に刺激を送ります。それを脳が合成して知覚することで，やっと音楽が再現されるのです。

時間ごとの音のエネルギー変化を表したものがいわゆる波形表示です。自然にある振動の本質は単振動と呼ばれるもので，波形表示すると綺麗なSIN波を描きます。波形を構成する点が中心軸から離れるほど音量が大きいことを示し，波の間隔が狭いほど周波数が高い，いわゆる高い音だということになります。SIN波を上下左右に伸び縮みさせると，音量，音程を変えられるわけです。

音色の発生

弦楽器は弦を振動させ，ボディと共鳴させることにより音を出します。管楽器はリードなどで発生した音を管内で共鳴させて音を出します。このような物理的に音を発生する楽器は，発音メカニズム上，振動体の固有振動数や共鳴の振動数を基本として，その整数倍の周波数の音成分を多く含んだ音色を発生します。また，振動する媒体の両端が固定されているか否かで，両者には音成分に違いが現れています。

図1



基本周波数に対して，その3倍音，5倍音といった奇数次の倍音を組み合わせると管楽器の基本波形ができあがります。

「基本音程に対する倍音」という考え方は音色を作るうえでの基本になります。単振動によって発生するSIN波は「ポー」という感じの柔らかい音ですが，これに高次倍音を増やすと「キーン」という金属的な音になっていきます。どんな倍音をどの程度含んでいるかで音色の基本性質が決定されているともいえるでしょう。

確かに楽器の種類により基本波形は違ってくるのですが，実験的に違う楽器から基本波形をとり出して連続的に合成したものを再生してみると，ほとんど音色を聞き分けられないというという事実に直面します。音量変化（エンベロープ）を除かれた音というのは識別しにくいものなのです。

結局，人間は音色を音の立ち上がり時(アタック）から減衰（リリース）までの変化やノイズなどの要因を加味して聞き分けていることがわかっています。

基本波形とエンベロープ，これが音色の2大要素です。シンセサイザなど，人工的に音色を合成する場合にもこの考え方が基本となります。

どうやって基本波形を作り，どのように変化させるか？ それが各種音源方式の違いです。そして，いかなる音源でも発声のメカニズムがわかっていれば，コンピュータ上でシミュレードすることができるはずなのです。

それでは実際にはどんなふうに音色が作られるのかをざっと見てみましょう。

アナログシンセサイザ

さまざまな周波数成分を含んだ信号から不要な成分を減算していってできるのがアナログシンセサイザサウンドです。

VCO（基本周波数発生）
VCF（フィルタ）
VCA（音量調整）

の3段階で構成されています。

フィルタというのは最近のシンセサイザの「音色を飾る」フィルタとはまったく違います。逆に不要なものをそぎ落とすのが主な役目です。フィルタなしのデジタルシンセはありますが，普通のアナログシンセにはフィルタなしは考えられません。

VCOは矩形波や鋸波など何種類かの基本波形を出力します。それにフィルタを組み合わせ望みの音を切り出し，最後にエンベロープを与えます。

周波数の減算もデジタルに行うことは可能ですが，フィルタの性能にも限界がありますから，特定の周波数とはいっても，その周辺の音を含んだ「分厚い音」になります。それがアナログシンセの特徴といっていいでしょう。逆に，アナログシンセ全盛の時代にはFM音源によるピュアなデジタルシンセサウンド（悪くいえば薄っぺらい音）は画期的なものだったのです。

いまさらアナログシンセという感じももっともですが，伝統の重みか，デジタル音源の出力にフィルタをかけてアナログ風に扱おうという楽器も少なくありません。

FM音源

お馴染みのFM音源。これはSIN波をSIN波で変調していくもので，大雑把にいって三角関数の演算で音を合成するものです。皆さんお馴染みですね。実際にどういうぐあいに合成されるのかについて詳しくは丹

氏の記事を見てください。最近は，なにもSIN波でなくてもいいじゃないか，というタイプもあるようです。

SIN波合成型

音が倍音の構成により規定されるなら，倍音合成で音を作ろうという発想によるものです。考え方は単純，倍音の比率を変えて基本波を作っていきます。KORGのDWGS音源などは自然音を分析したものを持っており，倍音合成で再現するかたちの音源です。加工もきわめて簡単に行うことができます。

詳しくは石上氏の記事を参考にしてください。

PCM音源

昔，メロトロンという楽器がありました。楽器の音を1音ずつテープに録音しておき，キーを押すとその力でテープが回転ヘッドに押し付けられ再生を始めるというものです。なにしろ鍵1個ずつにテープとヘッドがついているのですから，なかなか大変な機械です。

それがやがてもっと効率のいいサンプラーに代わり，PCM音源となります。PCM音源の考え方は単純です。メモリ上にあるデータを加工しながら再生していくだけです。もっとも単純なものは1波形分だけメモリに持ち，エンベロープなどは加工して作成します。

1音のアタックからリリースまでを持っているのが一般的なPCM音源です。音程の変化はデータを加工して再現します。

音域や強度による違いをリアルに出すために生まれたのがマルチサンプリングのPCM音源です。音域ごとにいくつかのデータを切り換えていきます。理想をいえば，1音ずつサンプリングするのが最高です。突き詰めれば結局はメロトロンと同じことになるのが面白いところです。

また，楽器音の最大の特徴とされるアタック音だけサンプリングして，シンセ音と合成するとか，サンプリング音の波形で変調を行うとか複合的な方式も多く存在します。

その他，サンプリング波形自体を時間に変化させれば波形カスケードというテクニックになります。KORGのAI音源では音域に影響されない成分を加味して音をリアルにしています。サンプリングの応用例は無限といえます。

AD PCMによる合成

PCM音源はデータがなんでもいいのですからもっとも柔軟な音源といえます。特に数学的に合成しやすいところに面白味があります。

X68000のAD PCM音源はPCM音源の1種ですが，データを差分のかたちで1/4に圧縮しているためハードウェア的な変調や合成はできません。そこでソフト的にPCM変換して加工を行わなければなりません。リアルタイムに行うことは非常に重い処理ですから今回の特集記事ではオフライン処理で加工を行っています。

かつて，1988年8月号「真夏の夜の数値演算」で加藤氏が同様なAD PCMによる音の分析と合成を行っていました。AD PCMとPCMへの変換がデシベル表現だったらしく対数値を用いる必要があったことがわからず，音量誤差が極端なノイズになって出ていました。

サンプルしたポイントからデータを補間してから近似関数をフーリエ展開するという，なかなかスマートな方法で，手書きの波形を合成したりできました。波形の近似が音の近似になると仮定していたところには問題があるかもしれません。結局，手書きの波形を入力するということ自体があまり意味がないことなのかもしれませんが（フェアライトCMIなどではできる）。手書きでは周波数成分を考慮することが極端に難しくなります。

そこで，今回石上氏が行っているような生のままのデータでのフーリエ解析です。どちらかといえば，これが普通のアプローチなのですが，フーリエ変換という重い処理が必要となってきます。なお，そこでは波形を再現するためにCOS成分を導入しています。SIN成分だけにすることもできるのですが，それでは再合成したものが元の波形とは似ても似つかなくなってしまいます。このCOSというのはなにを意味しているのでしょうか？

数学のグラフを見てもわかるように，COS波はSIN波の位相が90度分ずれたものです。SIN波とCOS波を聞き分けられる人が存在しないように，位相のずれは波形に大きな影響を与えますが音色にほとんど影響しないようです。音色が違えば波形は違いますが，波形が違うからといって音が違うとは限りません。

全然違う形の波形でも同じ音になることが多々あります。たとえば，図2に示された波形はどれも同じに聞こえる音（同じ周波数成分を持っている）の例です。

ですから波形から単純に音色を判断することはできそうでできないことなのです。波形表示というのは位相差のない状態にしてからでないと説得力に欠けます。

石上氏と同様に丹氏もFM変調のフィードバック部分で位相のずれを導入しています。これは実測されたFM音源の波形に基づくものなのですが，作用するのがモジュレータとしてですから，これは音色に影響があるのかもしれません。

ちなみに，ポスト“Walkman”として最近発表されたミニディスクやDCCというオーディオ機器でもPCMデータを圧縮しているのですが，これはAD PCMのような圧縮ではなく，なんとリアルタイムでフーリエ変換を行い，演奏に関係なさそうな部分を削っているようです。なかなか凄い世の中になったものです。

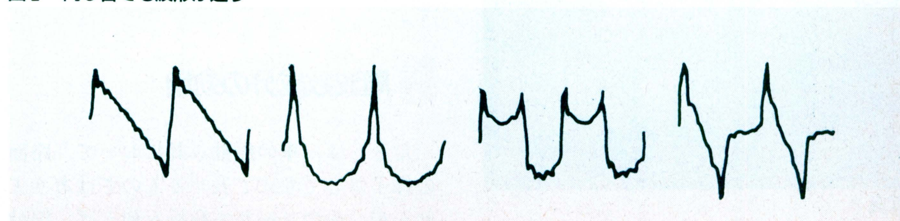
*　　*　　*

今回の特集はPCM（AD PCM）を使っての音色合成が中心ですが世の中はあなどれません。

FM音源の基本公式もベッセル関数を使って展開を繰り返せばSIN波の倍音合成式に変形できることが知られています。逆変換できれば，自然音を倍音構成に分解しFM音源で合成するということも可能なのかもしれません（不可能でも不思議はないが）。

しかし，とりあえずPCMの可能性は無限です。音色レベルの合成ができれば，1曲丸ごとというのも不可能ではありません。逆にこういった非合理的なことができるのもコンピュータならではといえるのではないでしょうか。

図2　同じ音でも波形が違う



FFT/逆FFTによる音声分析

冬の夜長のスペクトル解析

Ishigami Tatsuya 石上 達也

「音」というものを扱う際の常套手段がこのフーリエ解析です。音を周波数成分に分離し，そして周波数成分から音を合成するというプロセスを使うときわめて直感的に音色作りができるのです。フーリエ変換はほかの用途にも応用可能です。

音とはなんでしょう？ そう，小学生の頃に習ったとおり空気の波です。では，どんな波なのでしょうか？ 今回はここらへんからメスを入れて音の本質に迫ってみましょう。残念ながら，三角関数・級数などを理解している人を対象にしています。中学生以下の皆さんごめんなさい。

音の関数化

写真1を見てください。これはピアノの音を今回のプログラムを使って，視覚化したものです。目を細めて細部が見えないようにして全体像を見てください（モザイク除去じゃないって)。同じような模様が繰り返されていますね。で，この同じような模様の1ユニットのことを1波形といい，この1波形を再現するのに必要な時間を1周期といいます。また，1秒間に収まった波の数のことを周波数といいます。

写真2の波形を見てください。実際の音としてはトランペットのような音なのですが，なにかの図によく似ていますね。そう，y＝sin（t）のグラフとそっくりです。このように音を三角関数で関数化できれば，コンピュータで簡単に解析することができるようになります。

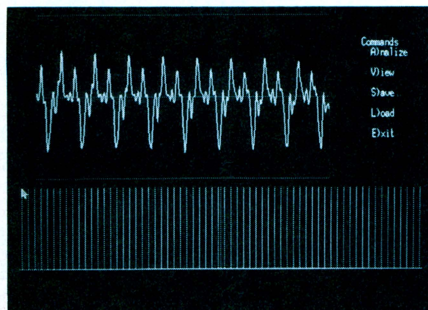


写真1　ピアノの音

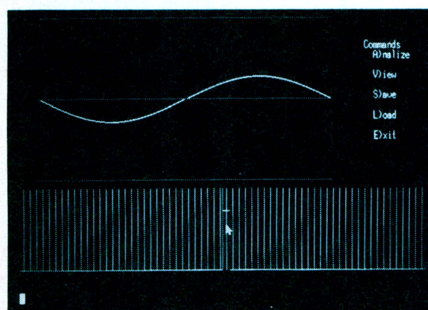


写真2

図2の波形は簡単に関数化できましたが，図1の波形をこのように関数化するのは結構難しそうですね。

フーリエ解析

その昔，フーリエという大変偉い物理学者がいました。彼は，自分の論文を書くのに，いままでの数学理論では足りずに，自分で新しい理論を作り出しました。その彼の理論によると，

「周期2πを持つような周期関数f(x)は，

$$f(x)=\frac{a_0}{2}+\sum_{k=1}^{\infty}(a_k\cos kx+b_k\sin kx) \qquad \text{(式1)}$$

のように変形することができる」

のだそうです。要するに，関数の種類を問わず，一定の周期で繰り返すものすべてが三角関数の和で表されるというのです。先ほどのピアノの音だって周期を持っているから三角関数化できるというのです。

なかなか疑わしい法則ですから，初めのうちは素直に信じられないでしょう。かの大天才数学者ラグランジュでさえも，非常に疑ったそうですから，この式が信じられなくても，無理はありません。しかし，浮き世の常として，我々がどう思おうと，この式は世の中で立派に通用しているようです。

周波数成分の効用

たとえば，車の振動設計において，路面の様子はとても式で表せるものではありませんが，先のフーリエ解析を使えば，振動状態は三角関数の和で表せます。三角関数の振動は実験室でも簡単に再現できますし（重心をわざとずらしたシャフトを回すだけ)，三角関数の和(三角関数そのものではない）については，「線形性」という性質が使えるので，どんな振動でも解析することができるようになります。

こうして，車にこのタイヤをつけるときは，サスペンションの硬さはこのぐらいにすべし，という値などが計算されるのです。

さらに，式1をよく見てください。sin，cosの項が整数倍に増えていっていますね。つまり，あの式は任意の複雑な波形を，一つの基本波とその整数倍波の和で表してしまうというものでもあるのです（このような動作をスペクトル分析といいます)。

今度は，路面ではなく車体自身の振動状態を調べて，このスペクトル分析を行います。この結果，人の脳や内臓の共振周波数である2～5Hz(この値はおおいに個人差がある）の周波数成分が多く含まれていると，それだけその車は酔いやすい車だということになります。

一般乗用車だと，この共振を避ける周波数帯はある程度広く採ってあるのですが，F1車のようなレーシングカーは，そんな悠長な設計をしていられません。とにかく，相手の車よりも速く走るためには，タイヤの空気圧やサスペンション系の調整は必須事項です。私はF1車の設計なんてやったことがないのですが，丹氏やU氏によるとAMIGAのレーシングシミュレーションゲーム「Indiana Police 500」では，そのあたりのセッティングは結構重要事項のようです。本物のF1ならなおのことでしょう。F1ドライバーの体重管理が厳重なのも，ここらあたりに要因がありそうですね。

また，フーリエ解析ではありませんが，日光をプリズムを使ってスペクトル分析すれば，「この波長の光が，この強さで含まれているということは，○○が燃えている証拠だ」と，誰も行ったことのないはずの太

陽の成分を解析することもできます（どーでもいいけど，どうやったら高校物理の教科書のような綺麗な結果が出るのでしょうね？）。

周波数成分を調べること（スペクトル解析）はなかなか応用範囲の広い手法なのです。

音をスペクトル解析する

音の重要な性質を定義するのに，周波数というものがあります。俗にいう，音が高い/低いというやつです。また，この他に音色という問題が残っています。たとえば，古村さんも私もカラオケで，同じ音程で「SAY YES」を歌えます（と信じたいです）が，2人の声を間違える人はいないでしょう。これは，2人の声の持っている音色が違うためで，音を波として考えた場合，これは，波の形，すなわち波形が違うことによるものです。

波形をそのまま数学的に解析するのは，非常に困難です。トランペットの口真似でsin波のみで歌を歌えるような人は例外として，たいてい人の声というのは，簡単に数式で表すことのできないような複雑な波形を持っているものです。コンピュータは定量的な問題の「どのくらい」を扱うのは得意ですが，定性的な問題の「どのような」を扱うのは不得手です。

さて，このようなときにも威力を発揮するのが，前述のスペクトル解析なのです。

これを使えば，複雑な波形の問題を倍音成分の重なり具合の問題に帰結させることができるようになるのは，先ほどお話したとおりです。というわけで，PCMデータをフーリエ変換にかけて各周波数成分ごとの強さを調べてみましょう。ただし，これだけでは面白くないので，ここで得られたフーリエ変換の結果を変えてみて，逆にどのような波形が合成されるのかも実験してみます。

その昔，NED（New England Digital）社から出ていたシンクラビアというシンセサイザやフェアライト社のCMIでは，この機能を使ってサンプラー機能を実現していました。ただし，シンクラビアの場合フーリエ変換を担当するのはDECのコンピュータで，フルオプションで1億円したそうです。CMIもマルチCPUの山のようなマシンでしたから，いかに大変かはおわかりいただけるでしょう。

逆フーリエ変換

（式1）における係数ak，bk（フーリエ級数という）をなんとか求めて周波数ごとの成分を求めます。ここで（式1）には∞というコンピュータにとって扱いづらい記号が出てきています。また，f(x)の範囲が［0，2π］というのも，ちょっと不便です。そこで，f(x)の範囲をTとし，サンプル数をN，サンプリング間隔をhとすると

$$T=N*h \qquad \text{（式2）}$$

となります。また無限級数の和をN項目までの和で打ち切ることにします。このとき（式1）は，次のように書き換えられます。

$$f(x)=\sum_{k=0}^{N}\left\{a_k\cos\frac{2\pi k}{N}x+b_k\sin\frac{2\pi k}{N}x\right\} \qquad \text{（式3）}$$

高校生の後半で習う式，

$$\sum_{m=0}^{N-1}\cos\frac{2\pi k}{N}m\cdot\sin\frac{2\pi l}{N}m=0 \qquad \text{（式4・1）}$$

$$\sum_{m=0}^{N-1}\cos\frac{2\pi k}{N}m\cdot\cos\frac{2\pi l}{N}m=\begin{cases}0\\N/2\end{cases} \qquad \text{（式4・2）}$$

$$\sum_{m=0}^{N-1}\sin\frac{2\pi k}{N}m\cdot\sin\frac{2\pi l}{N}m=\begin{cases}0\\N/2\end{cases} \qquad \text{（式4・3）}$$

を使って（これらの式は，教科書にはおそらく載っていないので，ちょっと厚めの参考書を調べてみてください），

$$a_k=\frac{2}{N}\sum_{m=0}^{N-1}y\cos\frac{2\pi k}{N}m \qquad \text{（式5・1）}$$

$$b_k=\frac{2}{N}\sum_{m=0}^{N-1}y_m\sin\frac{2\pi k}{N}m \qquad \text{（式5・2）}$$

が出てきます。これで，ようやく（式3）はコンピュータにかけられる形になりました。

Nの値が小さいときは，このままでもよいのですが，今回の音の波形のように，Nの値が大きくなると，このままではあまりにも時間がかかりすぎてしまいます。

ここで，うまい抜け道を考えた人たちがいます。これらの手法を総称してFFT（Fast Fourier Transform）と呼ぶのですが，今回はN＝512として，FFTのなかでもCooley-Tukey法を使って切り抜けることにします。

複素級数ckを，

ck＝(ak-ibk)/2　（式6）

とおくと，あまり自信はないのですが，フーリエ逆変換の式（5・1），（5・2）は

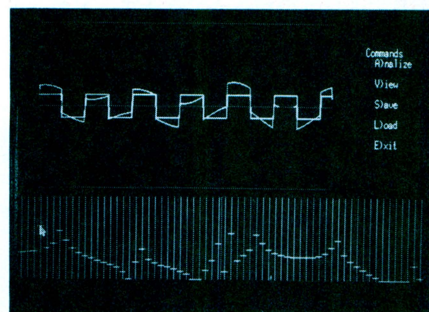


写真3　周波数の解析

以下のように変形できるそうです（あっ，exp(ix)＝cos(x)＋isin(x)は知っていますよ，ね）。

$$y_m=\sum_{k=0}^{N-1}c_k\exp\left\{i\frac{2\pi k}{N}m\right\} \qquad \text{（式7）}$$

また，与えられたサンプリングデータ列を偶数番目/奇数番目で分け，それぞれ，

ｓｍ＝ｙ2m　（式8・1）

ｔｍ＝ｙ2m＋1　（式8・2）

とおくと，それぞれのデータ列に対するフーリエ変換は，

$$S_k=\frac{2}{N}\sum_{m=0}^{N}s_m\exp\left(-i\frac{2\pi km}{N/2}\right) \qquad \text{（式9・1）}$$

$$T_k=\frac{2}{N}\sum_{m=0}^{N}t_m\exp\left(-i\frac{2\pi km}{N/2}\right) \qquad \text{（式9・2）}$$

のようになります。ここで，Sk Tkは，それぞれｓ，ｔに対するフーリエ級数です（それぞれの扱うべきサンプリングデータの数が半分になっていることに注目）。

また，Cooly-Tukey法ではこれらのあいだには以下の式が成立することが証明されています。

$$C_k=\frac{1}{2}S_k+\frac{1}{2}\exp\left(-i\frac{\pi k}{N/2}\right)\cdot T_k \qquad \text{（式10・1）}$$

$$C_{k+N/2}=\frac{1}{2}S_k-\frac{1}{2}\exp\left(-i\frac{\pi k}{N/2}\right)\cdot T_k \qquad \text{（式10・2）}$$

繰り返し，このようなデータ列の振り分けとこの漸化式を使うとサンプリングデータの数が半分ずつになっていき，やがてサンプリングデータは1個になります。ただひとつのデータに対しフーリエ級数を求めるというのは，結局その数そのものということになりますので（式10・1），（式10・2）を用いれば，フーリエ変換を用いずにフーリエ級数を求めることができるようになります。ここで，N＝512（＝29）に固

定したというのが生きてきます。Nが2のベキ乗で表せなければ，この方法は使えませんものね。

以上は，ほとんど参考文献1の受け売りですので，詳しいことを知りたい方はそちらを当たってみてください。

プログラムの設計

いきなりAD PCMデータを相手にするのは大変ですので，なんらかの方法でこれをPCMデータに変換する必要があります。また，再現性を高めるためにも，フーリエ変換にかける前に，きっちり1周期分を切り出す必要が出てきます。

これには，Oh!X　Mook第1段の「Z-MUSICシステム」（定価2,300円）に掲載されているZVT.Xというプログラムを用います。主著者の西川氏によると「西川さんが男であることを忘れて，愛してしまいそうになるほど素晴らしい本」だそうですのでアバンチュールな世界を知りたい人には必須アイテムです（と書くことを約束してしまった，ストリートファイターIIで彼に敗れた私)。くれぐれも，内容がバビンチョだからといって，ソフトバンクに抗議のFAXを送ったり，嫌がらせの電話をしないようにお願いします。

このPCMデータを切り出すプログラムは，ZVT.Xでなくてもよいのですが，その場合はPCMデータをshort型の16ビット数で出力するようにしてください。また，PCMデータの長さは512ポイント（short型なので1024バイト）にしておいてください。これ以上あっても無視されますし，足りなくても残りの領域に0を詰め込まれるだけですが，若干演算精度が落ちてしまいます。

いま，PCMデータのサンプリングレートを15.6kHzとすると，512ポイントでは，約0.03秒分のサンプリングデータを扱えることになります（(式2）においてT=0.03)。バッファを1周期分としてまるまる使った音というのがあったとしても，1/0.03=33.333［Hz］ですから，これはもう問題にするべき音ではないでしょう（人間の聞こえる最低の音が20Hzだそうです)。

基本波を可聴波とすると，40倍波の音(=40オクターブ上の音）なんてとても聞こえないので，とりあえず32倍波までを表示し，これらを変更可能とします（シンクラビアやCMIと同じ)。ただし，波形の再現性の問題から，内部ではきっちりN倍波までサポートすることとします。

また，これらの級数の表示形式ですが，とびとびの数値をとる傾向がありますので，常用対数をとってデシベル表示とすることにします。

使い方

操作スタイルとしては，キーボードからコマンドを入力しつつ，補助的にマウスを用います。コマンドは最初のアルファベット1文字を入力します。

まず，解析したいPCMデータ(X68000のPCMファイルではなく，先ほどお話ししたように，ZVTを用いて切り取った生のPCMデータ）をLoadコマンドでロードします。で，これを解析するのはAnalizeコマンドです。X68000に数値演算プロセッサが積まれている場合は，約8秒。そうでない場合は，約13秒で解析が終了します。

画面の中央部にレベルメーターが出ていますが，この位置がバラバラにずれたはずです。このレベルメーターはそれぞれの周波数の大きさです。左から，cosの係数が32個分，さらに赤い線をはさんで，32個分がsinの係数です。ただし，いちばん左のレベルメーターは，バイアス値といってcos 0（=1）の係数で，波形全体レベルを変えるときに使います。

このレベルメーターをマウスの左ボタンでドラッグ（?）して，任意の位置に持ってきます。細かい設定や画面に表示しきれないような大きな値を設定するときは，そのメーターをマウスカーソルが指しているときに右ボタンを押します。すると，画面やや下側に数値入力を促すメッセージが出てきますので，これに従って，数値を設定してください。

また，レベルメーターはすべて絶対値で表示されていますので，負の値を設定したいとき(つまり，その項の位相を180度ずらしたいとき）も同様の方法を使ってください。(式6)からもわかるように，今回のプログラムではsinの係数を我々の感覚とは符号の違う形式で表していますので，これが気にいらない方は上述の方法を使って，いちいち変換していくかプログラムを改造してみてください。

設定が終了したら，Viewコマンドを用いて，このフーリエ級数による波形を画面に表示します。表示された波形が気にいらないときは，以上の動作を繰り返します。気にいった（?）波形が合成できたら，そのPCMデータをSaveコマンドでセーブしておきましょう。終了はEXITコマンドです。PCMデータをX68000で鳴らしたいときには，前述のZVT.Xを用いて必要があれば適当に加工して，AD PCMファイルを作成してください。

発展性の考察と多少のホラと

今回のプログラムでは，波形をいじれるのでスペクトル解析の結果をsin，cosの項ごとに分けて扱っています。cos成分は，単にsin成分の90度位相の遅れた成分を表しているだけですので，純粋にスペクトル解析を行って，各周波数ごとの強さを調べてみたいときは，

$$A\sin(t)+B\cos(t)=\sqrt{A^2+B^2}\sin(t+\Phi) \quad \text{(式11)}$$

ただし $\Phi=\tan-1(B/A)$

を使って，プログラムを改造してみてください。

また，今回はバッファ内容をまとめて1周期とみなしていますので，うまくデータを切り出さないとかなり不正確なデータとなります（ぴったり2のn乗個の波でなければほとんど意味をなさない)。表示された波形からマウスで切り出して指定できるようにできればよかったですね。

今回のプログラムは音の解析をとりあえずやってみようというノリで開発したもので，あまり細かいところ，特にユーザーインタフェイスについては目をつぶってあります。直接AD PCMファイルを扱えたり，波形をいじりながらPlayコマンドなどで，すぐに波形の再生ができればよかったかもしれません。プログラムがあまり大きくなるのが嫌だったのと，時間が押していたので今回は見送らせていただきました。AD PCMからPCMへの変換やその逆の方法などは，参考文献5に載っていますので各自の自由課題としましょう。この方法ができれば，プログラムを抜けずに波形を再生することも簡単でしょうし，もっと実用的なツールを作ることもできるでしょう。あとは皆さんでがんばってみてください。

参考文献

1) 大滝，木下，小林共著 機械工学のためのFORTRAN入門，廣済堂
2) 大崎順彦 地震動のスペクトル解析入門，鹿島出版
3) 岸 恒行，トランジスタ技術1990年2月号，電子楽器の成り立ちとその変遷，CQ出版
4) 西川 他，Oh!X Books「Z-MUSICシステム」，ソフトバンク
5) 島田 他，X68000パワーアッププログラミング，アスキー出版局

リスト1

```c
#include	<basic0.h>
#include	<basic.h>
#include	<mouse.h>
#include	<graph.h>
#include	<fcntl.h>
#include	<math.h>

typedef struct cmplx {		/*  複素数型 */
	float	Re;
	float	Im;
} CMPLX;


#define		NASI	0x4e415349		/* 'NASI' */
#define		BUFF_SIZE		512

	CMPLX	c[BUFF_SIZE+1],
		y[BUFF_SIZE+1],
		z[BUFF_SIZE+1];

	double	s[64+1];
	short	data[BUFF_SIZE+1];
	short	moto[BUFF_SIZE+1];

	char	fileName[40];
	int	fn;

void		drawGraph(short *,int);
void		fft(CMPLX *,CMPLX *, int, int);


main() {
unsigned char		strtmp0[258];
unsigned char		key[1];

	int	i;
	int	ms_x,ms_y,ms_bl,ms_br;

	console(0,32,0);
	initScreen();
	for (i=0;i<=511;i++){
		data[i]=0;
	}

	box(30,30,30+512,30+256,8,NASI);
	line(30,30+128,30+512,30+128,8,NASI);
	mouse(4);
	mouse(1);
	msarea(6,300,699,428);

	while (1) {
		msstat(&ms_x,&ms_y,&ms_bl,&ms_br);
		mspos(&ms_x,&ms_y);
		if(ms_bl == -1) mslDown(ms_x, ms_y);
		if(ms_br == -1) msrDown(ms_x, ms_y);

		strncpy(key, b_inkey0(strtmp0), sizeof(key));/*リアルタイム一文字入力*/

		switch(toupper(*key)) {
			case 'A':
					analize();
					break;
			case 'E':
					mouse(0);
					wipe();
					exit(0);
					break;
			case 'V':
					view();
					break;
			case 'L':
					loadData();
					break;
			case 'S':
					saveData();
					break;
		}

	}
}

clrWin(int line) {
	int	cnt;

	locate(0,28);
	while(line--) {
		cnt = 60;
		while(cnt--) putchar(' ');
		putchar('¥n');
	}
}

/*
**  マウスの左ボタンが押された
*/
mslDown(int x, int y) {
	int	ch;

	ch=(x - 3) / 11;
	erasVr(ch);
	if(s[ch] >= 0) {
		s[ch]=(double)(428-y) / 30.0;
		s[ch]=pow((double)10.0,s[ch]);
	} else {
		s[ch]=(double)(428-y) / 30.0;
		s[ch]=-pow((double)10.0,s[ch]);
	}
	drawVr(ch);
}

/*
**  マウスの右ボタンが押された
*/
msrDown(int x,int y) {
	int	ch;
	int	tmp;

	ch=(x - 3) / 11;
	locate(0,28);
	if ( x < 353 ) {
	 printf("ＳＩＮ関数の%d番目の係数の値は%4.2fです¥n",
		ch+1,s[ch]);
	} else {
	 printf("ＣＯＳ関数の%d番目の係数の値は%4.2fです¥n",
		ch-31,s[ch]);
	}
	b_input("新しい値を入れて下さい－>",0x204,&tmp,-1);
	erasVr(ch);
	s[ch]=tmp % 0x8000;
	drawVr(ch);
	clrWin(2);
}

/*
**  画面の初期化
*/
int		initScreen() {
	int	i;

	screen(2,0,1,1);
	for (i=0;i<=31;i++){
		line(6+i*11,300,6+i*11,428,9,NASI);
		line(358+i*11,300,358+i*11,428,9,NASI);
	}
	line(351,300,351,428,5,NASI);
	locate(76,3);
	puts("Commands ");
	locate(77,5);
	puts("A)nalize");
	locate(77,7);
	puts("V)iew ");
	locate(77,9);
	puts("S)ave ");
	locate(77,11);
	puts("L)oad ");
	locate(77,13);
	puts("E)xit ");

	for (i=0;i<=63;i++){	/*ボリュームを描く*/
		s[i]=0;
		drawVr(i);
	}
}

/*
**  ボリュームを描く
*/
int		drawVr(int ch) {

	int	x,y;

	x = ch*11;
	if(s[ch] == 0)
		y = 428;
	else
		y = 428-log10(fabs(s[ch]))*30;

	if( y < 300) y = 300;
	if( y > 428) y = 428;
	box(x,y,x+11,y,9,NASI);
}

/*
**  ボリュームを消す
*/
int		erasVr( int ch) {

	int	x,y;

	x = ch*11;
	if(s[ch] == 0)
		y = 428;
	else
		y = 428-log10(fabs(s[ch]))*30;

	if( y < 300) y = 300;
	if( y > 428) y = 428;

	fill(x,y,x+11,y,0,NASI);

	line(6+ch*11,300,6+ch*11,428,9,NASI);
	if ( ch==31 | ch==32 ) {
		line(351,300,351,428,5,NASI);
```

```
	}
}

/*
**	画面に波形を描く
*/
void		drawGraph(short dat[], int clr) {
	int	oy,ny;
	int	i;

	oy=158-dat[0]/32;
	for (i=1; i <= BUFF_SIZE ; i++) {
		ny = 158 - dat[i] / 32;
		if(ny > 30+256) ny = 30+256;
		if(ny < 30)	ny = 30;
		line(i+29,oy,i+30,ny,clr,NASI);
		oy=ny;
	}
	box(30,30,30+512,30+256,8,NASI);
	line(30,30+128,30+512,30+128,8,NASI);
}

/*
** ファイルより読み込む
*/
loadData() {
	int	fn;
	int	i;

	int	length;

	locate(0,28);
	b_input("File Name: ",sizeof(fileName),fileName,-1);
	clrWin(1);

	fn = open(fileName, O_BINARY | O_RDONLY);
	if (fn >= 0) {
	 length = filelength(fn);
	 if (length > BUFF_SIZE * sizeof(short)) {
		read(fn, (void *)data, BUFF_SIZE * sizeof(short));
	 } else {
		read(fn, (void *)data, length);
		for(i=length/sizeof(short); i<= BUFF_SIZE; i++)
			data[i]=0;
	 }
	 close(fn);

	 fill(31,31,30+511,30+255,0,NASI);
	 drawGraph(data, 11);

	 for(i=0; i<= BUFF_SIZE; i++)
		moto[i]=data[i];
	} else {
	 locate(0,28);
	 puts("ファイルがオープン出来ません");
	}
}

/*
** ファイルに保存する
*/

saveData()
{
	int	fn;

	locate(0,28);
	b_input("File Name: ",sizeof(fileName),fileName,-1);
	clrWin(1);

	fn = open(fileName, O_BINARY | O_WRONLY | O_CREAT);

	if (fn >= 0) {
		write(fn, (void *)data, BUFF_SIZE * sizeof(short));
		close(fn);
	} else {
		locate(0,28);
		puts("ファイルがオープン出来ません");
	}
}

/*
**	波形を解析する
*/
analize() {
	int	i;

	locate(0,28);
	puts("Now Calculating ...");

	for(i=1; i <= BUFF_SIZE; i++) {
		y[i].Re = moto[i+1];
		y[i].Im = 0;
	}
	fft(y,c,BUFF_SIZE,-1);

	for(i=0; i<=31; i++) {
		erasVr(i);
		s[i] = c[i+1].Re;
		drawVr(i);
	}
	for(i=32; i<=63; i++) {
		erasVr(i);
		s[i] = c[i-30].Im;
		drawVr(i);
	}

	clrWin(1);

}

/*
**	波形を合成する
*/
view() {
	int	i;

	locate(0,28);
	puts("Now Calculating ...");

	for(i=1; i <= BUFF_SIZE; i++) {
		y[i].Re = data[i];
		y[i].Im = 0;
	}

	for(i=0 ; i<=31; i++)
		c[i+1].Re = s[i];
	for(i=32; i<=63; i++)
		c[i-30].Im = s[i];

	fft(c, z, BUFF_SIZE,1);

	for(i=1; i <= BUFF_SIZE; i++) {
		data[i] = z[i].Re;
	}

	fill(31,31,30+511,30+255,0,NASI);
	drawGraph(moto, 11);
	drawGraph(data, 13);

		clrWin(1);

}

/*
**FFT解析プログラム
**(参考文献1による)
** n:データの総数
** iw == -1 : フーリエ変換
**		y:標本値
** iw == 1 : 逆フーリエ変換
**		y:フーリエ級数
** c:出力
*/
void		fft(CMPLX *y, CMPLX *c, int n,int iw) {
	CMPLX	str;
	float	theta;
	int	i,is,j,k,m,max;

	for( i=1 ; i <= n; i++) {
		c[i].Re = y[i].Re;
		c[i].Im = y[i].Im;
	}
	for( i = j = 1 ; i <= n; i++) {
		if( i < j)  {	/* swap(&c[i],&c[j]) */
			str.Re = c[j].Re; c[j].Re = c[i].Re; c[i].Re = str.Re;
			str.Im = c[j].Im; c[j].Im = c[i].Im; c[i].Im = str.Im;
		}
		m = n / 2;
		while( j > m ) {
			j = j - m;
			m = m / 2;
			if(m < 2) break;
		}
		j = j + m;
	}
	max = 1;
	while(max < n) {
		is = max * 2;
		for( k = 1; k <= max; k++) {
		    theta = PI * iw * (k - 1) / max;
		    for( i = k; i <= n; i+= is) {
			j = i + max;
			/* str = c[j] * exp(theta) */
			str.Re = c[j].Re * cos(theta) - c[j].Im * sin(theta);
			str.Im = c[j].Re * sin(theta) + c[j].Im * cos(theta);
			c[j].Re = c[i].Re - str.Re;
			c[j].Im = c[i].Im - str.Im;
			c[i].Re = c[i].Re + str.Re;
			c[i].Im = c[i].Im + str.Im;
		    }
		}
		max = is;
	}
	if(iw != 1) {
	    for(i=1; i<= n; i++) {
		c[i].Re = c[i].Re / n;
		c[i].Im = c[i].Im / n;
	    }
	}
}
```

AD PCMの超活用

FM音源の波形を創る

Tan Akihiko 丹 明彦

AD PCMとPCMデータの交換さえできれば、コンピュータ上であらゆる音が作成できるはずです。ここでは手始めにAD PCMを使って皆さんお馴染みのFM音源をシミュレートしてみましょう。

今回は，FM音源の音をPCM音源で鳴らすことを試みる。

PCM音源は基本的に現実界の音をサンプリングして使うものである。しかし僕らは，PCM音源の処理するデータがデジタル化した波形そのものだということを知っている。外界からサンプリングしたデータだけしかPCMデータとして使えないわけではないのだ。FM音源の出力波形を計算機で合成してPCM音源に渡せば，PCM音源はその波形に忠実に従った音を出すことだろう。

デジタルは楽し

世の中デジタル音楽全盛。電子楽器で創造した音楽はいうにおよばず，クラシックや人間の声もデジタルレコーディングされ，CDに収められる。それが可能になったのは，PCM録音技術の発達のおかげである。もはや耳タコであろうが，音は空気の振動であり，その振動の波形をデジタル化してメモリに取り込めば音を記録することができる。逆にメモリから取り出した波形を空気の振動に戻してやれば音を再生することができる。これがPCM音源の原理である。

PCM音源は，途中にアナログ-デジタル変換(録音時)とデジタル-アナログ変換(再生時)が入るという意味で元の波形を完璧に保存しているわけではない。またこの点が熱心な音楽愛好家からCDなどのデジタル録音が嫌われる原因になってもいる。

が，サンプリング周波数をある一定の値より高くすれば，問題なく録音/再生ができる。これについてはサンプリング定理「ある周波数の音を完全に録音したいならばサンプリング周波数はその2倍とる必要がある。また2倍以上とれば完全に再生できる」というものがある。

たとえばCDのサンプリング周波数は約44kHzだが，これは人間の可聴範囲が約20kHzまでなので，その2倍強の値をとれば十分ということで決められている。アナログの音をデジタルにきっちり切ってしまっては誤差が出るのでは……という心配は無用である。完全といったら完全に再現できる。現にCDはもはや実験的試みでもマニアのための先進的録音/再生技術でもなく，ごくふつうの音楽メディアとして普及している。

それよりも，いったんデジタルのデータに落としてしまうことではかりしれないメリットが生まれることにこそ注目したい。デジタル情報は劣化しない。これはすなわち，録音した音声データの保存が楽であることを意味する。編集・加工も容易である。いろいろなフィルタを通しても劣化しないし，複数の波形を合成して新しい波形を合成することも簡単だ。アナログではきわめて難しかった問題の多くが解決される。デジタルは安価に高い再現性を出すことができるのである。

＊

FM音源もPCM音源もX68000に搭載されている音源なのだから，PCM音源でFM音源を真似るというこの試みは無意味のように思われるが，実はそうでもない。というのも，ここで音源のエミュレーションの手法を確立しておけば，理論的には，

・FM音源何声分でも演奏ができる(PCMだから「多重録音」をしてしまえばいい)

・X68000の音源をフルに使うOPMDなどをAD PCMのみで演奏できる。PCM8声の演奏だって簡単だ。

・ほかの音源のエミュレーションを行うことで，あらゆる楽器の演奏をAD PCMのみでこなせるようになる。オペレータを4つ以上持つFM音源やLA音源，X68000のAD PCMよりも高い機能を持つPCM音源など，けっこう高価なMIDI楽器を購入せずとも同じような音が楽しめる。また，実在しない楽器もシミュレートできる。オペレータ16個のFM音源で全オペレータにフィードバックを入れるとか……。

といったことがちょっとした波形合成プログラムを書くだけで実現してしまうのである。オーケストラだって合成できる。

ただし，それには数Mから数十Mバイトのメモリ（圧縮されているとはいえ，ディスク1枚でも3分弱にしかならない）とかなりのCPU時間を必要とするだろう……。

懸念される音質については，理論上，最大でS/N比73.8デシベル（データにはかなり制限があるが，そこらのカセットテープより上)，最悪時は限りなくS/Nが下がる。「AMラジオ程度」という表現はある程度妥当なところかもしれない。AMラジオでもオーケストラ演奏は再現できる（ある程度は)。X68000本体でサンプリングするのはなかなか難しいようだ。うまくサンプリングすれば結構いい音が出るはずなのだ。

実行環境

使用する言語はX-BASIC。コンパイルして実行することをすすめる。

特殊なハードウェアはいらない。ただし，メモリは十分に必要だ。AD PCMがサンプリング周波数15.6kHzで1秒間の波形を再生することを想定してプログラムを書いた(プログラム先頭の定数を書き換えればもっと長くも短くも録音できる)。ご存じとは思うが，PCMデータはメモリをかなり食う。もしBASICから実行して，「メモリが足りない」とBASICから叱られた方は，BASIC.CNFのフリーエリア設定の値をもっと大きくするか，現在5本取っているPCMトラックの数を減らして上手に使い回すかしていただきたい。

ここからは重要。このプログラムを実行するときにはzvt.xが必要だ。zvt.xとは，先日出版された本誌別冊で西川善司氏が発表したPCM-AD PCMサンプリング&ハンドリングツールである(とても便利で面白いツールなので使っていただきたい)。

PCM音源のデータは少しも特殊ではないのだが，X68000のAD PCMは，平たくい

えばPCMデータに圧縮をかけたようなものになっている。フォーマットが少し特殊なのである。このプログラムのBASIC側ではPCMデータしか合成しない。これをzvt.xに渡してAD PCMデータに変換してもらい，発音する。このためにzvt.xが必要なのだ。zvt.xを入手しないとこのプログラムは使いものにならないのだ。

FM音源の動作

FM音源を使って音楽を演奏するまでには，

1) 音色を設定する
2) MML(ミュージック・マクロ・ランゲージ：FM音源に与える「楽譜」に当たるもの)を解釈して発するべき音の高さと強さを決定する
3) FM音源が指定された音色と音の高さと音の強さで音を鳴らす
4) 2)と3)を次々に繰り返して音楽を演奏する

というプロセスを経ている。今回AD PCMで真似をしようというのは，このうち1)と3)である。つまり，与えられた音色パラメータと音の周波数(高さ)と音の強さを元にして，それに相当する音の波形を合成する(そのあとこれをファイルに格納し，zvt.xに渡す)。その準備として，FM音源の動作を知っておくことにしよう。

X68000に搭載されているFM音源はOPMと呼ばれている。FM OPERATOR TYPE-Mの略である。OPMの動作はX-BASICユーザーズリファレンスなどに載っているが，今回必要になりそうな知識を選んで簡単に紹介しておこう。

●発音数

OPMは8音まで同時に発生できる。これを，OPMは8つのチャンネルを持つということにする。

●オペレータ数

1チャンネルにつき4つのオペレータを使える。

つまりOPMは合計32個のオペレータを持っている(図1)。以下は1チャンネル分の回路の中にある4つのオペレータの話。

●オペレータ

オペレータは正弦波を発生する装置である(図2)。

●ピッチ

オペレータの入力のひとつであるピッチは音の高さを決める(図2)。

●エンベロープ

オペレータの出力にはエンベロープをかけて振幅を時間的に変化させることができる(図3)。たとえば音量を自由に変化させることができる。

●FM音源の波形合成

FM音源は，複雑な波形を出力するために，2つまたはそれ以上のオペレータを組み合わせて使う。

●波形の重ね合わせ

並列に組み合わせた2つのオペレータの出力はその合計である(図4)。

●変調

2つのオペレータを直列に組み合わせる

図1 FM音源(OPM)の構成

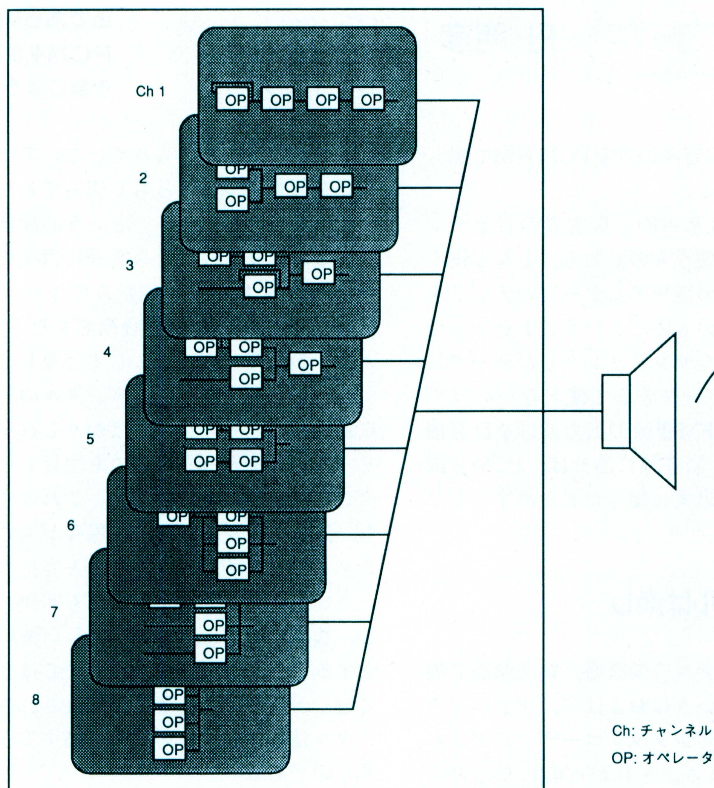


図2 オペレータ

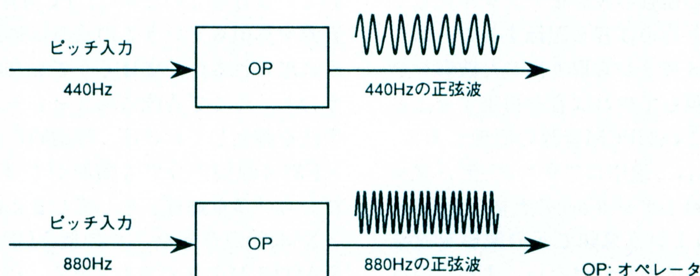
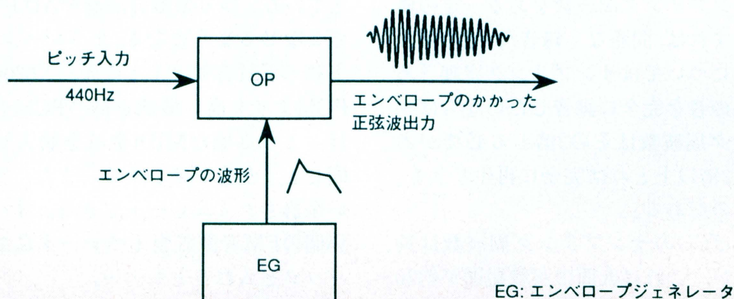


図3 エンベロープ



と，上流にあるオペレータがモジュレータ，下流のオペレータがキャリアという関係になる。キャリアはモジュレータからの入力を変調データとして受け取り，ピッチのデータに変調をかける(図5)。こうすることで複雑な波形を出力する。ピッチ，つまり周波数に変調をかけて目的の波形を作り出す，すなわち周波数変調(frequency modulation)が，FM音源という名称の由来である。

●フィードバック

4つのオペレータのうちのひとつはフィードバックできる。フィードバックとは，変調の一種だが，特にオペレータがその出力を自身の変調入力に戻して変調を行うものをいう(図6)。フィードバックを使うと，1オペレータで複雑な波形を出力することができる。フィードバックを行うオペレータにおいては，モジュレータもキャリアも自分自身ということになる。

●アルゴリズム

4つのオペレータをさまざまに組み合わせていろいろな種類の波形を作ることができる。この組み合わせ方をアルゴリズムと呼ぶ(図7)。

＊

これがほんの概要。これを完全に計算で真似るために，簡単なところから始めよう。

正弦波を作る

正弦波。サイン波ともいう。もっとも基本的な波である。波形を表す式は，

$y(t)=A\sin(2\pi ft+d)$

y:出力
t:時間
A:振幅
f:周波数
d:位相差

である(図8)。これだけの知識を元に，正弦波を合成してPCMデータに変換することを試みる(図9)。

先ほどAD PCMのサンプリング周波数を15.6kHzにすると決めた。そしてサンプリング時間は1秒にするとも決めた(1秒というと短いように感じるが，処理が重いので，たった1秒でもけっこう時間がかかるのだ)。1秒分，つまり，

15.6×1000×1=15600=ONESEC

個のサンプリング点を用意する(プログラム上は要素数15600個の整数の配列を用意する)。PCMデータは，16ビット符号つき整数なので，X-BASICの32ビット整数は冗長なのだ(メモリも2倍食う。ああもったいない)が，X-BASICにshort intがないので，以後これで処理する(ただしファイルレベルではzvt.xで処理するために16ビット整数に変換して格納している)。

ちなみに，基本波形を正弦波に限定する必要はない。1周期分，生楽器の波形を切り出して配列に入れれば，よくわからないくらい面白いことになる。

＊

16ビットPCM波形の振幅の範囲は，

$2^{15}=32768=$&h8000より，

図4　波形の重ね合わせ

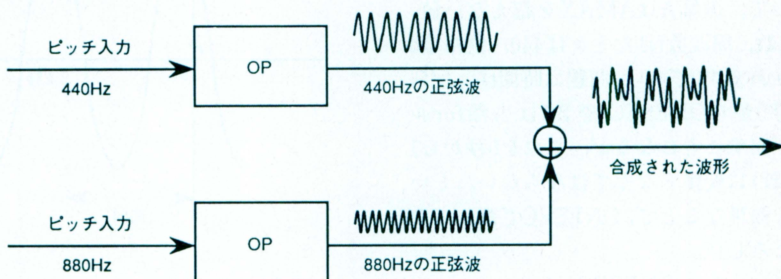


図5　周波数変調

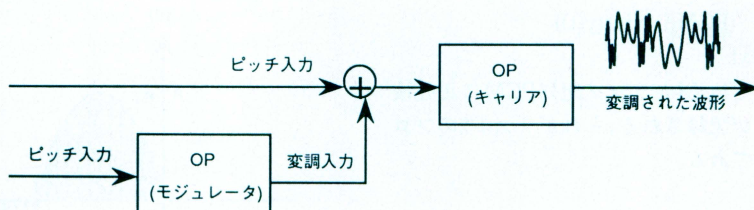


図6　フィードバック

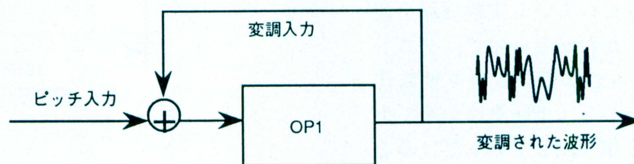
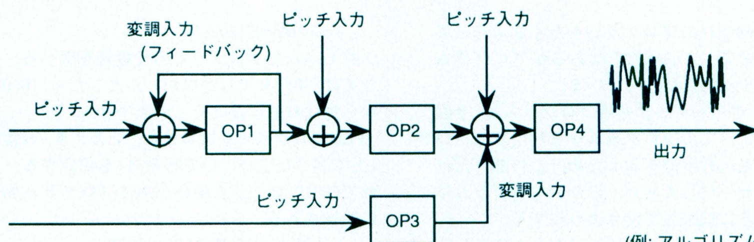
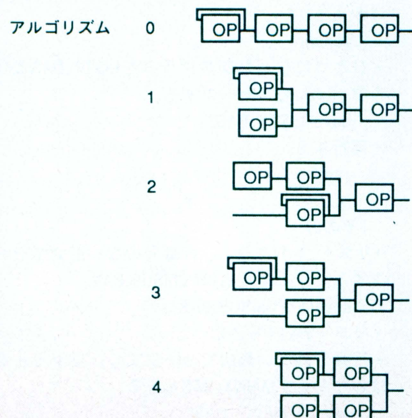
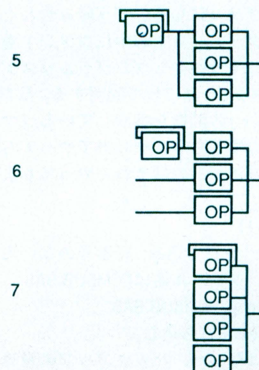


図7　アルゴリズム







$-32768<=y(t)<=32767$
である。プログラム中では振幅の最大値を定数AMAX＝&h8000としている。

＊

正直に正弦波をPCM化してみる。

$y(t)=A\sin(2\pi ft+d)$

において，振幅AはAMAXを超えない値，πは定数，周波数fはたとえば440，位相差dはとりあえず0でいい。問題は時間tだ。配列の添字0からONESEC(整数)は当然forループで回すことになるが，これを0秒から1秒(実数)に変換しなくてはならない。といっても簡単なことで，ONESECで割ってやればよろしい。

```
for i=0 to ONESEC-1
    t=i/ONESEC
    y(t)=Asin(2πft+d)
    PCM(i)=int(y(t))
next
```

これでPCMトラック配列PCMに正弦波の波形が記録される。それがリスト2のプログラムである。

整数化の話

ただ，このようにいちいち実数演算を使うのでは遅くてしかたがないので，リスト3以降のプログラムではできる限り整数化している。といっても，下駄をはかせて小数点以下の精度を作り出しているだけのこ

図8　正弦波

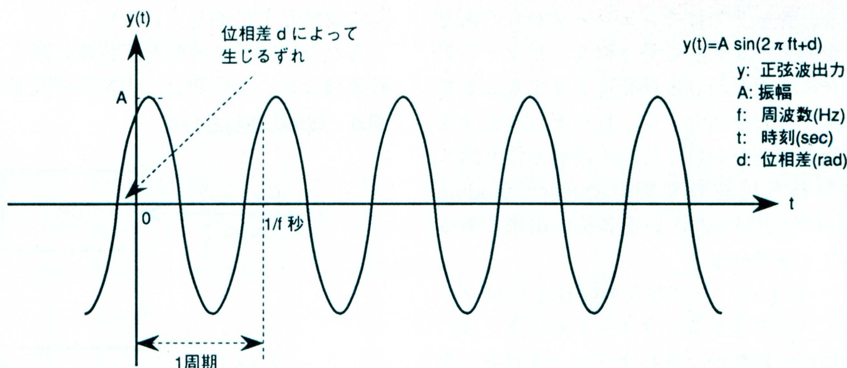
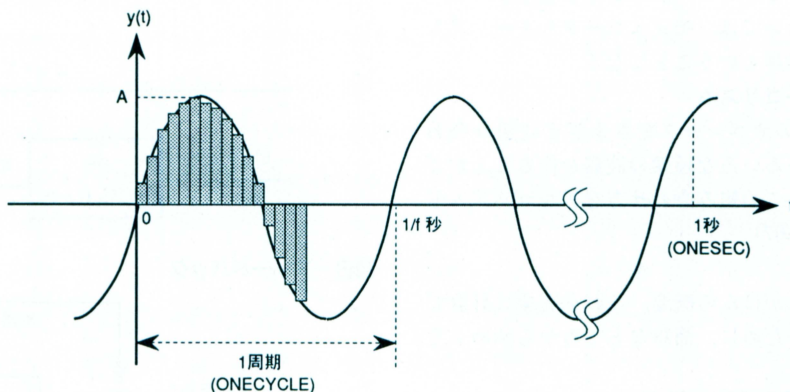


図9　正弦波をPCM記録する



プログラムの入力方法(BASICにおけるモジュール化のすすめ)

今回は細切れのプログラムをたくさん作ってしまったので，X-BASICではあるがプログラムの入力が少々面倒になっている。

リスト1のPCMデータ処理プログラムは，今回出てくるすべてのプログラムから利用する。いちいち打ち込む手間を省くため，この部分だけを独立にセーブしておき，ほかのプログラムリストの後ろに追加して使うようにする。

掲載したプログラムリストはどれも完結していない。いくつか組み合わせて初めて実行できるプログラムになる。どう組み合わせるかは適宜プログラムリストの中にコメントを入れてある。ここでは完全なプログラムリストの作り方を，リスト2を例にして説明する。なお，リスト2はリスト1の関数を参照している。2つの方法を挙げるのでどちらのやり方でやっても，また別のいい方法があればそれでやってもよい。

＊

(その1)

・リスト1を打ち込み，行番号のない形式でセーブする。ファイル名はPCMSUB.BAS。

save@"PCMSUB.BAS"

・リスト2を打ち込む。

・必要はないが，バックアップの意味でセーブする。ファイル名はとりあえずMAIN2.BASとする。

save"MAIN2.BAS"

・打ち込んだプログラムの行番号を調べる。たとえば230行までしかなかったとしたら，1000行から先は自由に使えると判断できる。

・PCMプログラムを読み込む。このとき，行番号(上で調べた空いている行番号)を指定する。これでPCMプログラム部分が本体プログラム部分に追加される。

load@"PCMSUB.BAS", 1000

・行番号の並びが悪いのがいやなら行番号を付けかえる。

renum

・ひとつになったプログラムをLIST2.BASというファイル名でセーブする。

save"LIST2.BAS"

・実行する。

run

＊

(その2)

・リスト1を打ち込み，行番号のない形式でセーブする。ファイル名はPCMSUB.BAS。

save@"PCMSUB.BAS"

・リスト2を打ち込む。

・行番号のない形式でセーブする。ファイル名はとりあえずMAIN2.BASとする。

save@"MAIN2.BAS"

・チャイルドプロセスでテキストエディタを呼び出す。メインプログラムとPCMプログラムを読み込んでおく。

!ed PCMSUB.BAS MAIN2.BAS

・両者をくっつける(この操作はエディタによって異なる)。

・ひとつになったプログラムをLIST2.BASというファイル名でセーブする。そしてBASICに戻る。

・残っていたプログラムを消去する。

new

・先ほど作ったプログラムをロードする。

load@"LIST2.BAS"

・実行する。

run

＊

全リストを通じて，同じ名前の関数は同じ内容，同じ動作にしてある。上手に打ち込む手間を省いていただきたい。

リスト1，3，5，7には関数だけを収めている。他のリスト2，4，6，8，9から呼び出されるだけで，それ自身では実行できないプログラムである。逆にいえば，リスト2，4，6，8，9にリスト1，3，5，7をインクルード(これはアセンブラやCの用語だが)したものが実行できるプログラムである。

とである。

たとえば三角関数の計算を節約するために，最初に1回だけ実数演算を使って三角関数テーブル(配列SinTableに格納する)を構成する。sin(x)の値は−1から1のあいだにあるから，それをAMAX倍すると値が−AMAXからAMAXのあいだになる。それを整数配列に格納し，以後はそちらを参照する。

参照するときに「このsin(x)はAMAX倍の下駄をはかせてある」ということだけ覚えておけば，実用上十分な精度が得られる。このテーブルの中身はファイル“SinTable.dat”に入れ，いつでも参照できるようにする。

細かいことをいえば，sin(x)は周期関数だがテーブルに格納した値は配列の添字内でのみ有効なので，当然周期はない。上手に添字を操作して，周期関数のように振舞わせる必要がある。このへんのことはアルゴリズムというよりテクニックの問題だから深入りはしない。原理的なものは本文で述べるので，プログラムではそれを整数化したものとして各自読みかえていただきたい。

エンベロープ

オペレータの出力を時間とともに変化させるエンベロープ。これの実現方法は簡単である。正弦波出力を，

$\sin(2\pi ft+d)$

とし(振幅は1とする)，エンベロープジェネレータ(EG)の出力を，

$E(t)$

とすると，オペレータからの最終的な出力は，

$E(t)\sin(2\pi ft+d)$

となる。掛け算すればいい。それだけ。

エンベロープのかけ方はわかった。次はエンベロープの作り方だ。OPMにおけるエンベロープは図10のようになっている。

楽器のキーが押されると(キーオンという)，エンベロープ曲線はその最大値に向かって上昇する。このときの上昇速度をアタックレイト(attack rate,AR)という。ARが大きいほど出力は急激に大きくなる。

いったん最大値まで昇りつめると，あとは下降曲線を描いていく。その下がり方には3段階ある。まず最大値からある一定の値まで落ちる。この値はディケイレベル(decay level,DR)と呼ばれる。出力がディケイレベルに落ちるまでの下降速度をディケイレイト(decay rate,DR)，または第1ディケイレイト(1st decay rate,D1R)という。

次に，キーオフ(キーが離されること)までの時間，別の下降速度で出力は下がる。この下降速度をサスティーンレイト(sustain rate,SR)，または第2ディケイレイト(2nd decay rate,D2R)という。このへんは用語が統一されていないようだ。今回のプログラムではSOUND PRO-68Kの記述に従ってDR,SRといっているが，X-BASICマニュアルやOPMそのもののマニュアルではD1R,D2Rと呼んでいる。

最後はキーオフから先の，出力が0になるまでの区間。この下降速度はリリースレイト(release rate,RR)。

これら出力の変化速度などをパラメータとしてエンベロープの波形を計算する。

今回いろいろと資料をあたったが，エンベロープの波形について明確な式での表現が見つからなかったので，半分以上は推測である(明らかに違うことはわかっているが，どう直せばいいのかわからない)。

変調

変調はFM音源のキモだ。ひとつでは単なる正弦波出力器でしかないオペレータが，コンビを組めば多彩で複雑な波形を生み出すことができるようになる。といってもその式は非常に簡単だ。

再び図5をご覧いただこう。変調のために直列に組み合わされた2つのオペレータは，上流にあるものをモジュレータ，下流にあるものをキャリアと呼ぶ。モジュレータの出力は，

$Em(t)\sin(2\pi ft)$

である。これに説明の必要はないであろう。Em(t)はモジュレータにかかるエンベロープである。さて，これをキャリアの変調入力に入れる。これは，先ほども述べたとおり，ピッチに影響する。周波数変調の要である。キャリアのエンベロープをEm(t)とすれば，キャリアの出力は次のようになる。

$Ec(t)\sin\{2\pi ft+Em(t)\sin(2\pi ft)\}$

式をよく見ると，sin｛ ｝の中にさらにsin( )が入っているのがおわかりいただけるだろう。まさに周波数変調なのである。

フィードバック

フィードバックは変調と似ているが，事態はもう少し複雑である。変調は，2つのオペレータが変調し，されるという関係にあったが，フィードバックの場合は両方ともそのオペレータ自身である。自分で自分を変調する。ひとつのオペレータで複雑な波形を出力できるので利用価値が高い。フィードバックの機能は，4つのオペレータのうちのひとつが持っている(オペレータ1)。

フィードバックの波形を求める式は，変調とほぼ同じだと思われる(このへんをはっきり書いた資料がないので，ある程度推測に頼っている)。今回は，

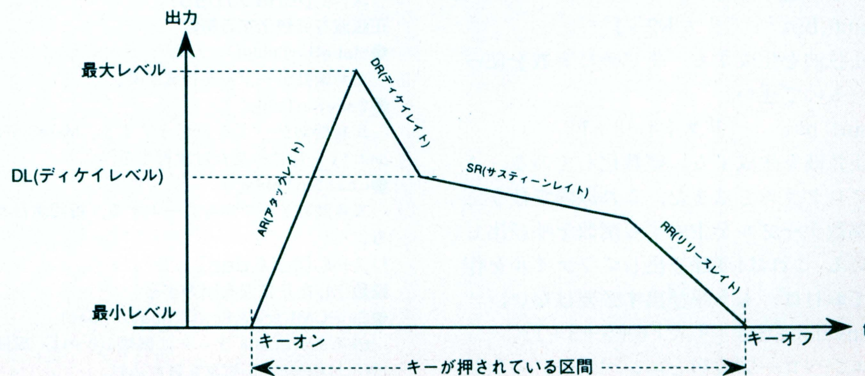
$E(t)\sin\{2\pi ft+d+E(t)F\sin(2\pi ft)\}$

という式に基づいて波形を作っている。E(t)はエンベロープ(2つのE(t)が共通であることに注意)，dは(まったくのデッチアゲだが)フィードバック回路がループを作っていることから遅れが生じていると推測して入れている。合計32個のオペレータを順番に処理しているのだから，ひとめぐりしてくるくらいの時間的遅れが生じてもおかしくはない(しかしデジタル回路だから遅れなんか出るわけないという主張のほうが説得力がある……)。

Fは変調度。これまた式がないので推測の域を出ないが，位相(sin｛ ｝の中身)をどれだけ変調させるかという文字どおりの解釈で通している。

こうして出てきた波形を見ると，1オペレータで相当に強烈な変調がかかっている。

図10　エンベロープのパラメータ



フィードバック恐るべし。

アルゴリズム

アルゴリズムはオペレータ間の信号の流れ方を決める。4オペレータのOPMの場合，8種類のアルゴリズムが用意されている。アルゴリズム2を例にとって，アルゴリズムの働きを見てみよう。ここでは4つのオペレータにそれぞれOP1,OP2,OP3,OP4と呼ぶ(X-BASICマニュアルやOPMマニュアルにはM1,C1,M2,C2と表記してある。モジュレータ1，キャリア1という意味である)。

・OP1はピッチ入力をもらい，フィードバックを経て変調データ1を出力する。

・OP2はピッチ入力をもらい，変調データ2を出力する。

・OP3は変調データ2をもらい，変調データ3を出力する。

・変調データ1と変調データ3を加算して変調データ4を得る。

・OP3は変調データ4をもらい最終的な音声波形を出力する。

もちろん，同時にエンベロープジェネレータなども作動している。

アルゴリズムはリスト9のプログラムで実現されている。文字どおり信号を順番に加工していっているだけである。アルゴリズムの記述は単なる力仕事で，前の項(フィードバック)までに覚えたことを組み合わせるだけである。

プログラムの使い方

プログラムの作り方は囲みを参照していただくことにしよう。申し訳ないが，作り方は多少ややこしい。しかし実行は簡単で，

run［return］

基本的にはこれだけ。あとはプログラムリスト中のパラメータをあれこれいじってみていただきたい。関数に渡すパラメータは囲みで解説してある。

sin0.bas　［リスト2+1］

：正弦波を生成する。まじめに実数を使っているので遅い。

sin1.bas　［リスト4+3+1］

：正弦波を生成する。整数化してある。このプログラムではまた，これ以降で使う三角関数テーブルを生成する関数を呼び出している。これは1度呼び出してファイルを作っておけば，もう呼び出す必要はない。

envelope.bas［リスト6+5+3+1］

：エンベロープ波形とエンベロープのかかった正弦波の波形を生成する。エンベロープ波形そのものを出力するために多少変なことをしている(気にする必要はない)。以後，正弦波出力(変調のかからないオペレータの出力)には，ここで使ったエンベロープを加味する正弦波出力を用いる。なお，純粋な正弦波を得たい場合には，エンベロープを矩形にする。具体的にはアタックレイトを最大の31に，ディケイレイトおよびサスティーンレイトを0に，リリースレイトは0でもいいがキーオフと同時に止めたいので最大の15にする。

mdltfdbk.bas［リスト8+7+5+3+1］

：このmdltfdbkというけったいな名前は，modulation&feedbackのハナモゲラである。変調とフィードバックをそれぞれ行った波形を出力する。

opmpcm.bas　［リスト9+7+5+3+1］

：以上を総合して，OPMのエミュレーションを行う。OPMの音色は，X-BASICの関数m_vget( )でプリセットの音色を吸い出している。今回掲載のプログラムの中でただひとつ対話的に動作する。ただしコンパイルすることが必要である(X-BASICインタプリタの仕様上，実行中にチャイルドプロセスを呼べない。zvt.xをプログラムの中から呼ぶにはコンパイルするしかなかった。それに，インタプリタでは実行が遅すぎるだろう)。

実行すると音色の番号を聞いてくるので，1から68(プリセットされている音色の番号)のなかから適当に選んで打ち込む。すると，その音色パラメータから波形を合成してPCMデータに直す(少し時間がかかる)。

終わると生成したPCMを鳴らすかプリセットされたOPMを鳴らすか尋ねてくるので適当なほうを打ち込む。何回でも聞き直せる。新しい音色に移りたいときは，ここで0と打ち込む。すると最初の音色指定に戻る。ここでもさらに0を打ち込むとプログラムは終了する。

関数リファレンス

引数の型は省略した場合は整数である。

リスト1［pcmsub.bas］：
PCMデータを処理する関数

●ClearPCM(trk)
trk番めのPCMトラックをクリアする(0で埋める)。

●FillPCM(trk,v)
trk番めのPCMトラックを値vで埋める。

●CopyPCM(totrk,fromtrk)
fromtrk番めからtotrk番めのPCMトラックへ転送する。

●AddPCM(totrk,fromtrk)
fromtrk番めのPCMトラックの内容をtotrk番めへ加算する。

●SavePCM(trk,filename;str)
trk番めのPCMトラックの内容をファイル名filenameでセーブする。

●LoadPCM(trk,filename;str)
trk番めのPCMトラックにファイル名filenameのファイルからロードする。

リスト2［list2.bas］：
正弦波の原理

●SinPCM0(trk,a;float,f;float,d;float)
trk番めのPCMトラックに振幅a(0〜1)，周波数f，位相差dで正弦波を書き込む。

リスト3［sinsub.bas］：
正弦波を整数化する関数

●MakeSinTable( )
三角関数テーブルを作成する。

●SaveSinTable( )
三角関数テーブルをセーブする。MakeSinTable( )と一緒に一度だけ実行する。

●LoadSinTable( )
三角関数テーブルをロードする。毎回実行する。

リスト4［list4.bas］：
整数化した正弦波を出力する

●SinPCM1(trk,a;float,f;float,d;float)
trk番めのPCMトラックに振幅a(0〜1)，周波数f，位相差dで正弦波を書き込む。

リスト5［evlpsub.bas］：
エンベロープの処理を行う関数

● Envelope(oct,note,Ke;float,AR,DR,DL,SR,RR,TL,KS)
エンベロープバッファに，指定された形のエンベロープ波形を生成する。oct,noteはキーコード。octはオクターブを，noteは音階を表す。Keはキーオンしている時間(0〜1)。

●SinPCM(trk,f;float,d;float)
trk番めのPCMトラックに周波数f，位相差dで正弦波を書き込む。エンベロープを参照する(以下，波形を生成する関数はエンベロープを参照する)。

リスト6［list6.bas］：
エンベロープの波形と，エンベロープのかかった正弦波の波形を出力する
関数はない(呼び出すのみ)。

リスト7［mdfbsub.bas］：
変調とフィードバックを行う関数

●Modulate(motrk,catrk,f;float)
motrk番めのPCMトラックをモジュレータに，catrk番めをキャリアにして，変調を行う。

●Feedback(trk,f;float,fdbk)
trk番めのPCMトラックに，フィードバックの出力を書き込む。

リスト8［list8.bas］：
変調とフィードバックを行った波形を出力する
関数はない(呼び出すのみ)。

リスト9［list9.bas］：
FM音源のエミュレーションを行う

●float frequency(oct,note)
キーコードoctおよびnoteからその音にあたる周波数を算出する。

●PCMvset_play(oct,note,k;float)
OPMの音色パラメータを用いてFM音源のエミュレーションを行う。kはキーを押している時間(0〜1)。音色はあらかじめOPMの音色を読み出す関数m_vset()を使って配列v(4,10)に読み出してあるものとする。アルゴリズムの判定や処理もこの中で行っている。

終わりに

FM音源の音の作り方は想像以上に複雑である。計算で出した音はなかなか本物と似ない。特に最後のプログラムは，実際の音色パラメータを使って波形を合成し，しかも合成したPCM波形と同じパラメータを使って演奏したOPMの音を聴き比べられるように作ってしまったので，全然違う音が出ることがばれてしまった。OPMの制御にはほかにもパラメータが必要だが，いろいろと省略してしまっている。あるいはそれが命取りなのかもしれない。

こうしてみると，PCM音源はあっけないほど単純でわかりやすい。加工も簡単である。ほとんど万能ではないかと思う。

参考文献

1) YM2151ユーザーズマニュアル，日本楽器製造株式会社
2) 小久保隆，誰にもわかるデジタル・サウンド入門，東亜音楽社発行・音楽之友社
3) SOUND PRO-68Kユーザーズマニュアル，シャープ
4) X-BASICユーザーズリファレンス，シャープ

リスト1

```
/*
/* List1: PCMデータのハンドリング
/*
/* サンプリング周波数 15kHz で 1秒間ぶんの波形を出力する設定
/*      (下の1行をメインプログラムに入れておくこと)
/*
/*int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
/*
func ClearPCM( trk;int )
  int i
  for i = 0 to LENGTH-1
    PCM(trk,i) = 0
  next
endfunc
/*
func FillPCM( trk;int, v;int )
  int i
  for i = 0 to LENGTH-1
    PCM(trk,i) = v
  next
endfunc
/*
func CopyPCM( totrk;int, fromtrk;int )
  int i
  for i = 0 to LENGTH-1
    PCM(totrk,i) = PCM(fromtrk,i)
  next
endfunc
/*
func AddPCM( totrk;int, fromtrk;int )
  int i
  for i = 0 to LENGTH-1
    PCM(totrk,i) = PCM(totrk,i) + PCM(fromtrk,i)
  next
endfunc
/*
func SavePCM( trk;int, filename;str )
  int fp, i, j=0
  /*
  for i=0 to (LENGTH-1)/2
    cPCM(i) = (PCM(trk,j) shl 16) + (PCM(trk,j+1) and 65535)
    j = j + 2
  next
  /*
  fp = fopen( filename, "c" )
  fwrite( cPCM, LENGTH/2, fp )
  fclose( fp )
endfunc
/*
func LoadPCM( trk;int, filename;str )
  int fp, i, j=0
  /*
  fp = fopen( filename, "r" )
  fread( cPCM, LENGTH/2, fp )
  fclose( fp )
  /*
  for i=0 to (LENGTH-1)/2
    PCM(trk,j  ) = cPCM(i) shr 16
    PCM(trk,j+1) = cPCM(i) and 65535
    if ( PCM(trk,j  )>=32768 ) then PCM(trk,j  ) = PCM(trk,j  )-65536
    if ( PCM(trk,j+1)>=32768 ) then PCM(trk,j+1) = PCM(trk,j+1)-65536
    j = j + 2
  next
endfunc
```

リスト2

```
/*
/* List2: 正弦波生成(原理)
/*
int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
ClearPCM( 0 )
SinPCM0( 0, 1/32#, 440#, 0# )  /* 440Hz, 最大振幅の1/32, 位相差0
SavePCM( 0, "sin0.pcm" )
/* チャイルドプロセス呼び出し(コンパイル時のみ有効)
/*system( "zvt -A sin0.pcm pcm" )
/* インタプリタ実行の場合(ファンクションキーを定義する)
print "SHIFT+F1 キーを押してください"
key 11,"!zvt -A sin0.pcm pcm@M"
end
/*
/* a;振幅, f;周波数, d;位相差
/*
func SinPCM0( trk;int, a;float, f;float, d;float )
  int i
  float t, a0
  a0 = a*AMAX
  for i=0 to LENGTH-1
    t = 1#*i/ONESEC
    PCM(trk,i) = int( a0*sin(2#*pi()*f*t+d) )
  next
endfunc
/********* ここに List1 (pcmsub.bas) をアペンドすること
```

リスト3

```
/*
/* List3: 正弦波を高速に生成するために整数化したテーブルを作る
/*
/* 1周期(=2π)ぶんの長さ ONECYCLE, 振幅 AMAX の正弦波のテーブル
/*      (下の1行をメインプログラムに入れておくこと)
/*
/*int SinTable(15599), ONECYCLE: ONECYCLE = ONESEC
/*
/* 波形テーブルの生成およびファイルへの格納(一度でよい)
/*MakeSinTable(): SaveSinTable()
/* 波形テーブルのファイルからの読み込み(毎回)
/*LoadSinTable()
/*
func MakeSinTable()
  int i
  float t0
  t0 = pi(2#)/ONECYCLE
  for i = 0 to ONECYCLE
    SinTable(i) = int( AMAX*sin(t0*i) )
  next
endfunc
/*
func SaveSinTable()
  int fp
  fp = fopen( "SinTable.dat", "c" )
  fwrite( SinTable, ONECYCLE, fp )
  fclose( fp )
endfunc
/*
func LoadSinTable()
  int fp
  fp = fopen( "SinTable.dat", "r" )
  fread( SinTable, ONECYCLE, fp )
  fclose( fp )
endfunc
```

リスト4

```
/*
/* List4: 正弦波生成(高速版)
/*
int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
int SinTable(15599), ONECYCLE: ONECYCLE = ONESEC
MakeSinTable(): SaveSinTable()  /* 1度実行すればよい(ファイルSinTable.datができる)
/*LoadSinTable()                /* 2回目からはこちら(ファイルSinTable.datを読み込む)
ClearPCM( 0 )
SinPCM1( 0, 1/32#, 440#, 0# )  /* 440Hz, 最大振幅の1/32, 位相差0
SavePCM( 0, "sin1.pcm" )
/*system( "zvt -A sin1.pcm pcm" )
print "SHIFT+F1 キーを押してください"
key 11,"!zvt -A sin1.pcm pcm@M"
end
/*
/* a;振幅, f;周波数, d;位相差
/*
func SinPCM1( trk;int, a;float, f;float, d;float )
  int i, t, a1, f1, d1
  a1 = int( a*AMAX )
  f1 = int( f )
  d1 = int( d*ONECYCLE/pi(2#) )
  for i=0 to LENGTH-1
    t = (i*f1 + d1) mod ONECYCLE
    PCM(trk,i) = PCM(trk,i) + a1*SinTable(t)/AMAX
  next
endfunc
/********* ここに List3 (sinsub.bas) をアペンドすること
/********* ここに List1 (pcmsub.bas) をアペンドすること
```

リスト5

```
/*
/* List5(evlpsub.bas): エンベロープ
/*
/* エンベロープを格納する配列
/*      (下の1行をメインプログラムに入れておくこと)
/*
/*int EVLP(15599), EMAX = &H8000
/*
func Envelope( oct;int, note;int, Ke;float, AR;int, DR;int, DL;int, SR;int, RR;int, TL;int, KS;int )
  int i, e, ke, tl, Rks, EMAX2
  float dB
  Rks = ((((oct shl 4) or note) shr 2) shr (3-KS))
  AR = AR*2 + Rks
  if ( AR > 63 ) then AR = 63
  DR = DR*2 + Rks
  if ( DR > 63 ) then AR = 63
  SR = SR*2 + Rks
  if ( SR > 63 ) then AR = 63
  RR = (RR*2+1)
  RR = RR*2 + Rks
  if ( RR > 63 ) then AR = 63
  dB = 0.75#*TL
  tl = int( EMAX * pow( 10#, -dB/10# ) )
  i = 0
  e = 0
  EMAX2 = EMAX*2
  if ( AR = 63 ) then AR = EMAX2 else AR = AR
  while ( i < LENGTH )
    e = e + AR
    if ( e >= EMAX2 ) then e = EMAX2
    EVLP(i) = e/2 * tl/EMAX
    i = i + 1
    if ( e = EMAX2 ) then break
  endwhile
  if ( DR = 63 ) then DR = EMAX2 else DR = DR
  if ( DL = 15 ) then dB = 93# else dB = DL*3#
  dl = int( EMAX2 * pow( 10#, -dB/10# ) )
  while ( i < LENGTH )
    if ( e <= 0 ) then break
    e = e - DR
    if ( e <= dl ) then e = dl
    EVLP(i) = e/2 * tl/EMAX
    i = i + 1
    if ( e = dl ) then break
  endwhile
  if ( SR = 63 ) then SR = EMAX2 else SR = SR
  ke = Ke * ONESEC
  if ( ke > LENGTH ) then ke = LENGTH
  while ( i < ke )
    if ( e <= 0 ) then break
    e = e - SR
    if ( e <= 0 ) then e = 0
    EVLP(i) = e/2 * tl/EMAX
    i = i + 1
    if ( e = 0 ) then break
  endwhile
  if ( RR = 63 ) then RR = EMAX2 else RR = RR
  while ( i < LENGTH )
    if ( e <= 0 ) then break
    e = e - RR
    if ( e <= 0 ) then e = 0
    EVLP(i) = e/2 * tl/EMAX
    i = i + 1
    if ( e = 0 ) then break
  endwhile
  while ( i < LENGTH )
    EVLP(i) = 0
    i = i + 1
  endwhile
endfunc
/*
/* エンベロープつき正弦波
/*    f;周波数, d;位相差
/*
func SinPCM( trk;int, f;float, d;float )
  int i, t, a, f1, d1, s
  a = AMAX
  f1 = int( f )
  d1 = int( d*ONECYCLE/pi(2#) )
  for i=0 to LENGTH-1
    t = (i*f1 + d1) mod ONECYCLE
    s = SinTable(t)*EVLP(i)/EMAX
    PCM(trk,i) = a*s/AMAX
  next
endfunc
```

リスト6

```
/*
/* List6: 正弦波生成(エンベロープつき)
/*
int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
int SinTable(15599), ONECYCLE: ONECYCLE = ONESEC
int EVLP(15599), EMAX = &H8000
/*MakeSinTable(): SaveSinTable()
LoadSinTable()
/*
/* エンベロープ
/*
Envelope( 0, 0, 0.5#, 12, 8, 1, 2, 5, 16, 0 )
SinPCM( 1, 0#, pi(0.5#) )  /* 周波数0, 位相をπ/2 ずらす→エンベロープの波形
SavePCM( 1, "evlp.pcm" )  /* 音はしない(zvt.xで読み込んで波形を見ることができる)
Envelope( 4, 10, 0.5#, 12, 8, 1, 2, 5, 16, 0 )  /* キーコード(4,10)は440Hz
SinPCM( 2, 440#, 0# )  /* エンベロープのかかった正弦波の波形
SavePCM( 2, "sinevlp.pcm" )
/*system( "zvt -A sinevlp.pcm pcm" )
print "SHIFT+F1 キーを押してください"
key 11,"!zvt -A sinevlp.pcm pcm@M"
end
/********* ここに List5 (evlpsub.bas) をアペンドすること
/********* ここに List3 (sinsub.bas) をアペンドすること
/********* ここに List1 (pcmsub.bas) をアペンドすること
```

リスト7

```
/*
/* List7(mdfbsub.bas): 変調(modulation), フィードバック(feedback)
/*
/* OPMのシステムクロック(約3.5MHz)
/*      (下の1行をメインプログラムに入れておくこと)
/*
/*int CLOCK = 3579545
/*
/* 変調
/*
func Modulate( motrk;int, catrk;int, f;float )
  int t, a, f1, s
  t=0
  a = int( ONECYCLE/pi(2#) )
  f1 = int( f )
  for i=0 to LENGTH-1
    s = SinTable(t)*EVLP(i)/EMAX
    PCM(catrk,i) = a*s/AMAX
    t = (t + f1 + PCM(motrk,i) + ONECYCLE*2) mod ONECYCLE
  next
endfunc
/*
/* フィードバック
/*
func Feedback( trk;int, f;float, fdbk;int )
  int t, a, f1, delay, s, fdbk1
  float fdbkf
  a = AMAX
  f1 = int( f )
  if ( fdbk = 0 ) then fdbkf = 0# else fdbkf = (1 shl (fdbk-1))/16#
  fdbk1 = int(fdbkf*ONECYCLE/2#)
  delay = 32*f*ONECYCLE*2/CLOCK
  t=0
  for i=0 to LENGTH-1
    t = (i*f1) mod ONECYCLE
    s = SinTable(t)*EVLP(i)/EMAX
    /*t = ((t + delay + fdbk1*s/AMAX) + ONECYCLE*4) mod ONECYCLE*/
    t = ((t + fdbk1*s/AMAX) + ONECYCLE*4) mod ONECYCLE
    s = SinTable(t)*EVLP(i)/EMAX
    PCM(trk,i) = a*s/AMAX
  next
endfunc
```

リスト8

```
/*
/* List8: 変調とフィードバック
/*
int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
int SinTable(15599), ONECYCLE: ONECYCLE = ONESEC
int EVLP(15599), EMAX = &H8000
int CLOCK = 3579545
/*MakeSinTable(): SaveSinTable()
LoadSinTable()
/*
/* 変調
/*
Envelope( 4, 10, 0.8#, 31, 0, 0, 0, 15, 16, 0 )
SinPCM( 0, 440#, 0# )
Modulate( 0, 1, 440# )
SavePCM( 0, "modulation.pcm" )
/*
```

```
/* フィードバック
/*
Envelope( 4, 10, 0.8#, 31, 0, 0, 0, 15, 16, 0 )
Feedback( 1, 440#, 3 )
SavePCM( 1, "feedback.pcm" )
/*
/*system( "zvt -A modulation.pcm pcm" )
/*system( "zvt -A feedback.pcm pcm" )
print "SHIFT+F1, SHIFT+F2 キーを押してください"
key 11,"!zvt -A modulation.pcm pcm@M"
key 12,"!zvt -A feedback.pcm pcm@M"
end
/********* ここに List7 (mdfbsub.bas) をアペンドすること
/********* ここに List5 (evlpsub.bas) をアペンドすること
/********* ここに List3 (sinsub.bas) をアペンドすること
/********* ここに List1 (pcmsub.bas) をアペンドすること
```

リスト9

```
/* List9: OPM 音源を PCM 音源でエミュレートする
/*    ※コンパイルして実行してください
/*    (チャイルドプロセス呼び出しがインタプリタでサポートされていないため)
int PCM(4,62399), cPCM(31199), LENGTH = 15600, ONESEC = 15600, AMAX = &H8000
int SinTable(15599), ONECYCLE: ONECYCLE = ONESEC
int EVLP(15599), EMAX = &H8000
int CLOCK = 3579545
char v(4,10)   /* OPM音色設定レジスタ
/*MakeSinTable(): SaveSinTable()
LoadSinTable()
char vo, q
str mml
m_init()
m_alloc( 1, 1000 )
m_assign( 1, 1 )
while 1
  input "音色番号(1-200; 0で終了します): ", vo
  if ( vo = 0 ) then break
  m_vget( vo, v )
  PCMvset_play( 4, 10, 0.8# )  /* キーコード(oct=4,note=10)は440Hz
  SavePCM( 0, "fm.pcm" )
  m_init()
  mml = "@"+str$(vo)+"o4 v12 q7 A4"
  m_trk( 1, mml )
  while 1
    print "0: 次の音色を試す"
    print "1: 合成したPCM音を鳴らす"
    print "2: 見本のOPM音を鳴らす"
    input "  選んでください: ", q
    if ( q = 1 ) then system( "zvt -A fm.pcm pcm" )
                /* このプログラムに限りコンパイル専用とする
    if ( q = 2 ) then m_play()
    if ( q = 0 ) then break
  endwhile
endwhile
end
/*
/* キーコードからの周波数の算出
/*  oct:オクターブ, note:音階
func float frequency( oct;int, note;int )
/*  str NT(14)={"C#","D","D#","-","E","F","F#","-","G","G#","A","-","A#","B","C"}
  int nt(14)={1,2,3,0,4,5,6,0,7,8,9,0,10,11,12}
  float f
  f = (32.7#/2#)*(1 shl oct)*pow( 2#, nt(note)/12# )
  return ( f )
endfunc
/*
/* 音色セット & 波形生成
/*  音色パラメータ配列 v(4,10) の詳細は m_vset() に準ずる
func PCMvset_play( oct;int, note;int, k;float )
  int i, alg, fb, op(4)
  int mt, dt1, dt2
  float f, f1(4)
  f = frequency( oct, note )
  for i=1 to 4
    mt = v(i,7): dt1 = v(i,8): dt2 = v(i,9)
    if ( mt = 0 ) then f1(i) = f*0.5# else f1(i) = f*mt
    switch dt2
      case 0: break
      case 1: f1(i) = f1(i)*1.41#
              break
      case 2: f1(i) = f1(i)*1.57#
              break
      case 3: f1(i) = f1(i)*1.73#
              break
    endswitch
  next
  for i=0 to 4: ClearPCM(i): next
  alg = v(0,0) mod 8
  fb = v(0,0)/8
  for i=1 to 4: op(i) = (v(0,1) and ((1 shl (i-1)))): next
  switch alg      /* アルゴリズム
    case 0: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            Modulate( 1, 2, f1(2) )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            Modulate( 2, 3, f1(3) )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 3, 4, f1(4) )
            CopyPCM( 0, 4 ):break
    case 1: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            SinPCM( 2, f1(2), 0# )
            AddPCM( 2, 1 )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            Modulate( 2, 3, f1(3) )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 3, 4, f1(4) )
            CopyPCM( 0, 4 ):break
    case 2: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            SinPCM( 2, f1(2), 0# )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            Modulate( 2, 3, f1(3) )
            AddPCM( 3, 1 )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 3, 4, f1(4) )
            CopyPCM( 0, 4 ):break
    case 3: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            Modulate( 1, 2, f1(2) )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            SinPCM( 3, f1(3), 0# )
            AddPCM( 3, 2 )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 3, 4, f1(4) )
            CopyPCM( 0, 4 ):break
    case 4: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            Modulate( 1, 2, f1(2) )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            SinPCM( 3, f1(3), 0# )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 3, 4, f1(4) )
            CopyPCM( 0, 2 )
            AddPCM( 0, 4 ):break
    case 5: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            Modulate( 1, 2, f1(2) )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            Modulate( 1, 3, f1(3) )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            Modulate( 1, 4, f1(4) )
            CopyPCM( 0, 2 )
            AddPCM( 0, 3 )
            AddPCM( 0, 4 ):break
    case 6: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            Modulate( 1, 2, f1(2) )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            SinPCM( 3, f1(3), 0# )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            SinPCM( 4, f1(4), 0# )
            CopyPCM( 0, 2 )
            AddPCM( 0, 3 )
            AddPCM( 0, 4 ):break
    case 7: Envelope( oct, note, k, v(1,0), v(1,1), v(1,4), v(1,2), v(1,3), v(1,5), v(1,6) )
            Feedback( 1, f1(1), fb )
            Envelope( oct, note, k, v(2,0), v(2,1), v(2,4), v(2,2), v(2,3), v(2,5), v(2,6) )
            SinPCM( 2, f1(2), 0# )
            Envelope( oct, note, k, v(3,0), v(3,1), v(3,4), v(3,2), v(3,3), v(3,5), v(3,6) )
            SinPCM( 3, f1(3), 0# )
            Envelope( oct, note, k, v(4,0), v(4,1), v(4,4), v(4,2), v(4,3), v(4,5), v(4,6) )
            SinPCM( 4, f1(4), 0# )
            CopyPCM( 0, 1 )
            AddPCM( 0, 2 )
            AddPCM( 0, 3 )
            AddPCM( 0, 4 ):break
endfunc
/********* ここに List7,5,3,1 をアペンドすること
```

MMLによる音楽表現法

Z-MUSIC公式ガイドブック

Nishikawa Zenji 西川 善司

音楽というものを扱うとき，頼りになるのが譜面の情報です。しかし，これがクセ者です。ここではZ-MUSICシステムのMML表記を使用してさまざまな音楽表現への対応法をまとめてみましょう。

ハロー。みんな元気かな。ZMUSIC.Xは手に入れたかな。今月発表となったZMUSIC.Xですが趣味に走りすぎたため全貌をつかんでもらえるまで少々時間がかかるかもしれません。ZMUSIC.Xに関する質問があれば随時答えていくつもりですのでなにかあったら葉書や封書でOh!X編集部「ZMUSIC.X係」までお便りくださいね。私が愛と誠意をもって答えちゃいます。

譜面→MML

多くの人は楽譜を見ながら打ち込んでいくことでしょう。オリジナル曲を創るのではなくてゲームミュージックやその他のコピーをやっている人にとってはいちばん役に立つアイテムかもしれません。本当なら譜面は参考にとどめておくほうがよい（楽譜の過信は禁物です）のですが，今回はあえてこの辺りをテーマに掲げてみました。ズバリ「譜面に記載された特殊記号をどんなMMLに直すのがベストか」をZMUSIC.Xを道具としてこれから皆さんと一緒に見ていくことにします。

装飾音符

図1を見てください。ジャズの金管パートやピアノ，ゲームミュージックではシンセリードのメロディパートに図のような記譜をよく見かけますね。この斜線の入った小音符は装飾音符(ornaments)と呼ばれるもので主音符（大きいほうの普通の音符）の音長からごく短い時間をもらってその音階で演奏するという意味です。まさに味付け的な存在です。

これは人間的なノリを醸し出すには絶好の技法ですがMMLでももちろん実演可能です。図1のような例でしたら，

E4<D32C8..

のようになります。また，曲調によっては，

E8..<D32C4

のようになったりします。

また，この例では装飾音符の音長を32分音符にしていますが，これも状況によって変わってきますから臨機応変に適切なものを選びましょう。

演奏する楽器によっては主音符のアタックを柔らかくグリッサンド（後述）のように演奏したい場合があります。こういう場合は装飾音符と主音符の間にスラー(タイ)を入れて，

E4<D32&C8..

のようにすればOKです。ただしこれはFM音源パートのみに有効です。理由はZMUSIC.Xマニュアルの24ページに譲りますが，これと同じことをMIDIでやるためには以下のようにします。

E4<D32&@B-1365D8..@B0

ピッチベンダーを使ってDの音を2度下げるわけです。主音符の発音後ピッチベンダーを中央に戻すことを忘れると後ろの演奏がおかしくなるので注意が必要です。ちなみにいまの例はベンダーレンジが1オクターブのケースです。

ジャズなどのピアノソロなどでは装飾音と主音の音量が重要です。ミスタッチ（誤って違う鍵盤を叩いてしまう）のようなニュアンスを出したい場合は装飾音の音量を主音の音量より小さくしてやるとよいでしょう。

E4<@U-20D32@U+20C8..

という感じです（FM音源の場合はベロシティはありませんから単純にボリュームコマンドに置き換えます)。

ところでピアノの場合はポルタメントのようなことをすると不自然になりますから先ほど示した装飾音符と主音符を滑らかにつなぐような技は不要かつ禁物です。

逆に装飾音を強調したいときもありますから，そういった場合は上の例でいけば2つのベロシティコマンドの順序を入れ替えてやります。

図2のような例もよくみかけます。こういった装飾音符が2つ以上の場合も，

G4B32<D32C8.

図1
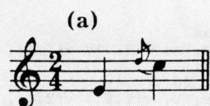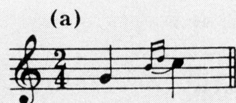


図2



ベンダーあれこれ

MIDI規格のピッチベンダーは14ビットパラメータですから－8192～＋8191までの値を取ることになります。

ZMUSIC.Xでは基本的にベンドレンジを1オクターブとしていますから1半音は，

8192÷12＝682.666≒683

2度（2半音）は，

8192÷12×2＝1365.333≒1365

ということになります。ですから，

@B683C

はC＋で，

@B1365C

はDの音程で演奏されるわけです。

FM音源でよく使われる，

C&D&E

のような演奏をMIDIでやるためには，

@B0C&@B1365C&@B2731C

のようにすればよいことになります。

しかし，楽器によってはベンドレンジが1オクターブでないものもあります。楽器のマニュアルや楽器の表示ディスプレイで確認しましょう。

KORG M1の工場出荷状態のプリセット音色のベンドレンジをすべて1オクターブに変更したファイルがZMUSIC.Xのディスクに収録されています。M1ユーザーはぜひご利用ください。

のように小音符を微小音長に置き換えるだけです。

このほか，装飾音符が後ろにきたりするものもあります。

トリル

ストリングスパートにおいて図3のような表記を見ることがあります。これはトリル(trill)と呼ばれる奏法で普通主音の2度上の音とを交互に高速に反復して演奏する技法でいってみればビブラートのようなものです。主音の2度上でないケースの場合もときどきあり，その場合は図4のように括弧の中に小音符を記述してあります。

図3をFM音源トラックでやるには，

L32｜：7C&D&:｜C&D

のようにします。“C&D”をズラーと並べるのもいいのですがあまり美しくないのでこの例では繰り返しコマンドを使っています。MIDIの場合は前出の装飾音符の例と同様にピッチベンダーを駆使して，

L32｜：7@B0C&@B1365C&:｜@B0C&@B1365C@B0

のようにします。いちばん後ろに“@B”を付けないと後ろの演奏がおかしくなってしまうので要注意です。

装飾音符のときと同様，微小音長の音長は状況によって違ってきます(上の例では32分音符)。余りにも高速のトリルを要求されるときはピッチモジュレーションを代用したほうがいいでしょう。

図3



図4
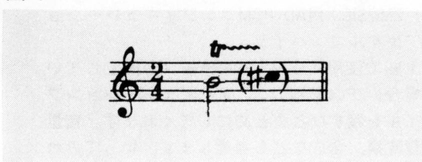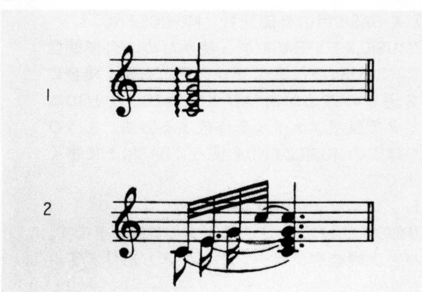


図5



アルペジオ

図5と図6はアルペジオ(arpeggio)と呼ばれるものです。一見すると単なる和音ですが和音の前に縦の波線が入っているのがミソです。これも装飾音の一種で和音を構成する1音1音を順番に高速に弾き，最終的には和音を奏でるという技法です。アコースティックギターのニュアンスやピアノのダンパーペダルの表現に適しています。

図5は低い音から高い音のほうへ，図6のタイプは逆に高い音から低い音のほうへキーオンしていきます。

さて，ZMUSIC.Xではこの奏法をコマンドひとつで表現可能です。和音コマンドのディレイを使うのです。図5は，

O4'C2EG<C',6

となり，楽譜で表記すると図5-2のようになります。図6は，

O5'C>GEC',6

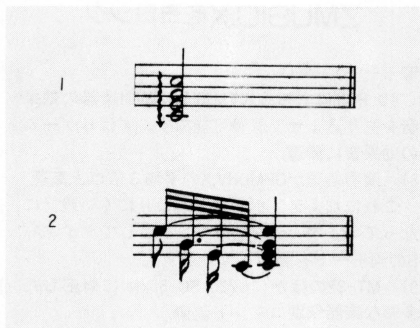
で楽譜では図6-2のようになります。ディレイの値は1音1音を発音していくタイミングに相当しますから，曲調や状況によって適切な値を選んでください。

また，シンセサイザやピアノなどはダンパーペダルを用いてこのニュアンスを出すことができます。ZMUSIC.Xでは@Dコマンドを用いて（図5の例なら），

L32 O4 @D1 CEG<C4.&C @D0

のようにします。

図6



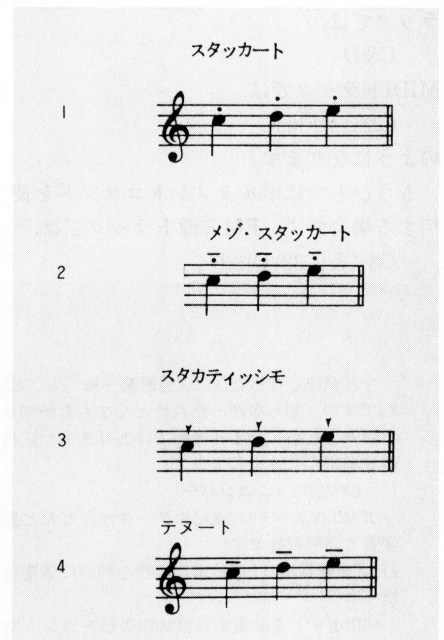
スタッカート

図7-1～図7-3を見てください。これらの記号はすべて発音長に関するものです。この記号は案外軽視されがちですが曲をシャキッとしたメリハリのついたものにするためには忘れてはならない演奏表記です。これはQコマンドを駆使して実行しますが大体の目安は以下のとおりです。

スタッカート(staccato)　Q4

メゾ・スタッカート(mezzo staccato)　Q6

図7



よく見掛ける演奏法の表記

バンドスコアやクラシックの譜面で五線の上にアルファベットでなにやら書いてあることがあります。よく登場するものをピックアップしてみます。

＊　＊　＊

●pizz.

ピチカート(pizzicato)と呼ばれるものでバイオリンなどのストリングスパートに記載されます。これは弦を弓でこすって音を出すのではなく指で弾いて音を出す演奏法です。MIDI楽器の場合はSTRINGSのグループにひとつの音色としてストアされていることでしょう。ちなみにベースやギターでは通常奏法がピチカートに分類されます。

さて，これに対して，

●arco

というのは通常の弓でこする演奏法の指定です。pizz.から元に戻るときなどに指定されます。

●harm.

ギターパートやストリングスによく見られます。これはハーモニクス(harmonics)といって弦を軽く指で押さえて倍音を強調して発声させる奏法です。別名フラジオレット(fraggioret)ともいいます。本来発声される音よりも高音を出すときによく行われます。音符の上に「harm.」あるいは◇を書いてこれを指定しますが，ほとんどの楽譜は実際に発音する音階を記してあるのでコンピュータミュージシャンの場合は無視して構わないでしょう。

＊　＊　＊

その他，ハンマリングオン(hammering on)やプリングオフ(pulling off)，チョーキング(choking)の指定がよく登場しますがたいてい実際に発音すべき音符が記載されているのでこれもコンピュータミュージックをやっている人には無関係でしょう。

スタカティッシモ(staccatissimo)　Q2

このほか，発音長に関する表記にテヌート(tenuto)がありますがこれらはQ7～Q8に相当します(図7-4)。

タイ，スラー，レガート

タイ(tie)はこれによって結ばれた同音程の2音の音長を足し合わせた音長で1音符として演奏するものです。これはZMUSIC.Xでは'&'を用いるのは常識ですね。これに似たものにスラー(slur)というのがあります。これは異なった音程の2音を切れ目なく演奏するもので表記はタイと同じです(故意に変えてあるものもありますが)。

さて，スラーをMMLで表すにはいくつかの方法が考えられます。ひとつは装飾音符のところで示した方法です。FM音源トラックでは，

C&D

MIDIトラックでは，

C&@B1365C

のようになります。

もうひとつはポルタメントコマンドを使用する場合です。FM音源トラックでは，

C8..&(C32,D)&D4

MIDIトラックでは，

C8..&(C32,D)&C4

のようになります。また，非常にアタックの弱い音の場合は以上のようなことを一切せず単にQ8で，

Q8CD

でそれっぽい効果を得られる場合もあります。

また，ある範囲を全体的にスラーのような感じで演奏するという意味でレガート(legato)という表記をした楽譜もあります。

ポルタメント

図8-1はポルタメント(portamento)と呼ばれるもので滑らかに音程をスライドしていく技法です。似たような表記にグリッサンド(glissando)というものもありますが(図8-2)，広義的には同一のものとして扱われるようです。

ZMUSIC.Xではこれまたコマンド一発で表記可能です。図8-1でしたら，

(G2<G)

のように記述します。図8-2も同様で構わないのですがどうしても表記にこだわりたいのなら装飾音符のときのテクを使って，

L16G&A&B&<C&D&E&F&G

…(FM音源トラック)

L16@B0G&@B1365G&@B2731G&@B3413G&@B4779G&@B6144G&@B6827G&@B8192G@B0

…(MIDIトラック)

のようにします。でもちょっと見にくいですねぇ。
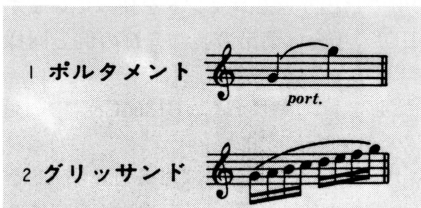
終わりに

結局，オリジナルなら自分の感性，コピーなら自分の耳がいちばん信頼できるものです。楽譜が間違っていたり変だったりするのはよくあること。あまり譜面を過信せずアバウトでいきましょう。

参考文献

1) 遠藤三郎，音楽用語の知識，株式会社日音
2) KB・SPECIAL '90 11月号，立東社

図8



ZMUSIC.Xをヨロシク

今月発売となったOh！X別冊第一弾「ZMUSIC.X」ですが，知らなかった人やどんなものか知りたい人のために（手前味噌ではありますが）特徴を列挙してみましょう。

1) OPMDRV.Xにコンパチ
OPMDRV.X＋OPMD.X用のデータがほとんど無変更で演奏可能です。

2) FM音源，AD PCM，MIDI楽器を同時に演奏可能
MIDIボードを装着すればMIDI楽器をコントロールすることができます。MIDIもAD PCMもOPM感覚で操作可能です。

3) 汎用トラックを80本持ち，その内最大32トラックを同時演奏可能
しかも1トラックで和音を8和音まで記述/発声できます。

4) 独自のAD PCMドライバを搭載
AD PCM発声時のプチというノイズを抑制。処理も高速化。

5) AD PCMコンフィギュレーションファイル用コマンドの装備
AD PCM発声時の音量/音程変更や合成が可能でそれらを各音階に割り当てることが可能。

6) ポルタメント，オートベンド，ピッチモジュレーションなどがFM音源，MIDI音源に対して使用可能
FM音源とMIDI楽器の操作体系をほとんど統一しました。MIDI楽器に対しては上記のほか，さらに怪しい機能も……。

7) 任意の時間に演奏中の任意のトラックに任意の演奏データを割り込ませて演奏可能(効果音モード)
AD PCMはもちろんFM音源やMIDI楽器の効果音を割り込ませて演奏可能です。ずばりゲームの効果音に最適。

8) 演奏処理がOPMDRV.Xの平均3倍以上高速
これに加えテンポずれが起こりにくい設計になっています。テンポカウンタとしてタイマA/Bの両モードを装備しています。

9) MT-32のほかにU220/SC-55/MIに対応した多彩な楽器個別コマンド装備
音色はもちろんドラムセットの変更などもできます。また，Rolandのエクスクルーシブに対応したコマンドもあるので上記以外のRoland系楽器のユーザーも安心(？)です。

10) MML演奏データをオブジェクトレベルのデータに変換可能
一般の音源ドライバのようなコンパイル機能があります。

11) 60種類のファンクションコールを装備
ZMUSIC.Xに装備されている命令すべてが機械語レベルで操作可能です。

12) 内部データフォーマットやワーク，ソースリストを公開
周辺アプリケーションの作成が楽です。さらにライセンスフリーですので特にことわりなしで商的利用もできます。

13) 業務用ADコンバータによってサンプリングされた2Mバイトにおよぶ AD PCMデータ群
これが目当ての人が多いかも……。バスドラムはちゃんとバスドラムしてます。

14) MIDI楽器の設定や音色などを吸い取る機能を装備
X68000側で楽器のデータを管理できます。これで楽器のメモリを書き換えちゃうような曲も安心安心。

*　*　*

このほかにいくつかのツールやZMUSIC.X周辺プログラムが収録されています。

15) トータルサンプリング「ZVT.X」収録
自分でサンプリングをしたり加工したりするツールです。オートサンプリング機能や4レベルの自動データ生成，波形表示機能などを装備。いままで，既存のAD PCMを利用する側に回っていた貴方も明日からサウンドクリエーター。

16) ZMUSIC.X用AD PCMコンフィギュレーションファイルコンパイラ
1曲で使用するAD PCMデータが決定している場合，ZPCNV.Xにコンフィギュレーションファイルを渡すひとまとめにしてくれます。音量音程変換，合成なども考慮します。いってみればコンフィギュレーションファイルコンパイラです。

17) X-BASIC用の外部関数「MUSICZ.FNC」
ZMUSIC.XをX-BASICから使うための外部関数です。ZMUSIC.Xは音楽プログラムを書く場合には2通りの方法がありひとつはED.Xなどのエディタで直接ファイルを作成する方法。もうひとつはこのMUSICZ.FNCを使ってBASIC上で書く方法です。

18) プレイヤー「ZP.X」
ZMUSIC.X用のデータを演奏するコマンドです。デバッグ機能やジュークボックスもあります。

〔Z-MUSIC LIVE SPECIAL　SHINDO ON STAGE〕

ファイナルラップ2より　

エンディング曲

ターボアウトランより　

KEEP YOUR HEART

Shindo Noriyuki 進藤 慶到

Oh！X LIVEでお馴染みの進藤君の協力を得て，Z-MUSICシステムのデバッグと改良は進められてきました。ここではデバッグの過程で作られたミュージックデータを挙げながらZ-MUSICシステムの概要を見ていきましょう。

祝！ Z-MUSIC完成

いやぁ，めでたいですね皆さん。なにがめでたいってそりゃあ，Z-MUSICが完成したことですよ。思えば私たちは今日まで限界を感じながらもひたすらOPMDRV＋OPMDを使い続けてきました。

確かに，このシステムでもそれなりの苦労をすればかなりのことが実現可能ではあるのですが，リストが大きくなる，他人にはなにをやっているのかよくわからない，処理が重いなどといった症状が常につきまといます。最小限のMMLコマンドだけを備えたOPMDRVでは辛くなってきたのです。

多くの人が，そろそろ新しい音源ドライバが必要だなと感じていたでしょう。しかしそれにはLIVE　inなどで蓄積されたOPMファイルとの互換性の問題があり，実現は難しいだろうなと私は思っていました。そこへZ-MUSICの登場です。「大幅に拡張されたMMLに加えて従来のデータがほぼ変更なしで演奏可能」，これを聞いて私はちょっと興奮してしまいましたよ。ときおり巷のいろいろなドライバの性能を横目で見ては，「OPMDRVにもあんなことができたらなぁ」と悔しさにも諦めにも似た感情を持った人は多いと思いますが，これからはZ-MUSICで思う存分MMLを書こうじゃありませんか。皆さんはもうZ-MUSIC持ってますか？ 持ってない人はいますぐ手に入れてシステムに組み込みましょう。

さて，今回載っているプログラムは（ディスクに収録されているツインビーもそうですが）私がZ-MUSICの「バグ出し」の際に作ったものでして，制作にあたっては当然ドライバをいじめるような使い方をせねばなりません。

あるときはトラック数をやたらめったら増やしてみたり，あるときはモジュレーション地獄を体験させてみたり，またあるときは和音地獄で攻めてみたり(「怪しい使い方」と呼ぶのだそうだ。本当は「正しい使い方」のはずなのに……)。そのためにリストには多少無駄な部分があるかもしれません。サンプルとして使っていただくことが決まってから少しは修正して見やすくなったんですけどね（いままでの私のプログラムと比べると随分ましなのでは。それにしても先月号のオーダインは自分が見てもひどい)。

サンプル曲の説明

プログラムの説明をしましょう。まずはナムコのカーレースゲーム「ファイナルラップ２」よりエンディングテーマです。ずいぶん古いゲームのようですが，多人数プレイの絶妙なバランスがウケてまだあちこちのゲームセンターに置いてありますね。ゲームはあまりやらないけどファイナルラップだけは好きだという人も大勢いることでしょう。

曲はというと，どうやらエンディングは２種類あるらしくて(あまりよく知らない)そのうちの一方ですが，まぁ聴いてもらえればわかると思います。とても爽やかな曲で，CDのアレンジバージョンと原曲をミックスしたような感じに仕上げました。心配していたサイズも多重ループやモジュレーションコマンドのおかげでトラック数の割には小さくなりました。

また，和音を多用していて演奏すると結構重いのですがなかなか正確なテンポをキープしていますし，発音ズレもありません（ここが音源ドライバのいちばん重要なところ！)。なお演奏にはX68000のほかにCM-64が必要です。ほとんどMIDIがメインで内蔵音源は飾りといった感じになってしまいましたが。

もうひとつの曲はセガのカーレース（これぱっか）ゲーム，ターボアウトランより「KEEP YOUR HEART」です。このゲームはあの有名な「アウトラン」の続編として出ましたがいまいちぱっとしませんでした。期待していたサウンドもプレイ中にはほとんど聞き取れずイライラしたものですが，CDを聴いてビックリ。音楽はやたらかっこイイ。元祖アウトランとはイメージががらっと変わってロック調の曲が多く，しかも名曲を連発していた頃のセガのパワーが感じられて１回で気にいってしまいました。一目惚れならず一聴惚れです。

今回作った曲もとてもよいので機会があったら一度CDを聴いてみてください。

おおもとのプログラムはOh!Xに投稿するためにOPMDRV上で作られていました。それをZ-MUSIC用にコンバートしたのですが，それによってMMLサイズはかなり縮まりました。LFOとポルタメントをサブルーチンで展開するという例のパターンをメロディとコードにまで使っていたので小さくなって当たり前なのですが，やっぱり小さいことは素晴らしい（しかも曲が始まるまでの時間がずいぶん短縮されている！)。

テクニック云々についてはオーソドック

スなものなので詳しく述べたりしませんが，ディストーションギターの音には苦労したので聴いてやってください。いい忘れるところでしたがMIDIは必要ありません。

これらのリストの入力には，付属のカウンタの値に十分注意してください。プログラムを入力したはいいが，いくら見直しても曲がズレてくるといったトラブルはなくなると思います。

ZMSファイルのすすめ

ところで，プログラムは2つともBASICではなくエディタ上で記述されています。従来ならOPMファイルと呼ばれた形式ですが，Z-MUSICシステムではZMSファイルと呼ばれます。投稿作品でも今後こういったかたちのものが増えるかもしれませんし，BASICしか使ったことがない人はこの際ですからエディタの使い方を覚えてしまいましょう。

ZMSファイルを直接エディタで書く利点というと，

1) 挿入が楽

行番号がないので，私のようにあとから追加修正しながら作る人には助かります。

2) コピー機能が使える

MMLのように同じところが多いと便利。

3) 配列変数を使わないでよい

音色設定やエクスクルーシブデータの送信など，BASICでは一度配列に数値を設定する必要がありますが，そんなことをせずに済みます。リストもすっきりします。

4) 周辺プログラムがすぐ使える

エディタを抜ければOSですからこれから増えるであろう周辺プログラムがすぐに使えます。

＊　　＊　　＊

挙げればきりがないですがざっとこんな感じです。少なくとも，エディタで書いていて煩わしさを感じることはありませんでした。

Z-MUSICをいろいろいじってみて私が感じたことは，汎用ドライバもこれくらいの機能がなくてはならないなということでした。世にはさまざまなドライバが出回っているようですが，それらと同等以上の機能を備えているといえます。ゲームに組み込めばBGM作成も楽になるでしょう。もちろんLIVE inに投稿される曲の完成度も上がるでしょう。やっとまともな地盤ができたなという感じがします。本来ならば標準で付いてくるドライバがユーザーの声を取り入れてバージョンアップされることが望ましいと思うのですが……。

最後に，私のわがままな要望に対して(バグレポートよりも要望のほうが多かったりしました) 素早く対応してくださったプログラマの西川善司氏に感謝します。

図1

```
final lap 2
/-- counter --
/  1:00001C44 00000000   2:00001B78 00000000   3:00001CF9 00000000   4:00001C57 00000000
/ 25:00001D11 00000000  26:00001D04 00000000  31:00001D04 00000000   9:000019F8 00000000
/ 10:00001C38 00000000  11:000019F8 00000000  12:00001FF9 00000000  13:00001C38 00000000
/ 14:00001C39 00000000  15:00001C44 00000000  16:00001C43 00000000  21:00001C44 00000000
/ 22:00001C38 00000000  23:00001FF9 00000000  24:00001C38 00000000   5:00001C39 00000000
/  6:00001B78 00000000  30:00001C38 00000000   7:00001AB8 00000000

keep your heart
/-- counter --
/  1:00000030 00007B60   2:00000031 00007B60   3:00000030 00007B60   4:00000030 00007B60
/  5:00000031 00007B60   6:00000030 00007B60   7:00000030 00007B60   8:00000048 00007B60
/  9:00000060 00007B60
```

リスト1

1-a

```
===============  FINAL.CNF  ==================
1=flanges_.pcm,v80,p6
2=brts.pcm,v65,p3,m1
.00c =fck.pcm,v95
.00d =shps_.pcm,p4,m2,v70
.00e =clp808.pcm,v130
.erase 1
.erase 2
```

1-b

```
===============  FINAL.ZMS  ==================
.comment -FINAL LAP 2- ENDING (C)namco  Programmed by ENG
/ 1991-08-13
/ for ZMUSIC.X
/ MIDI MODULE : CM-64
/-----------------------------------------------------------
/ TRACK SETUP
(i)
(b0)              / Base Channel = Internal
            / ベースチャンネルの設定はなるべく付けて下さい
/ Internal
(m1,2000)(aFm1,1)
(m2,2000)(aFm2,2)
(m3,2000)(aFm3,3)
(m4,2000)(aFm4,4)
(m25,2000)(aFm5,25)
(m26,2000)(aFm6,26)
(m31,2000)(aFm7,31)
(m9,2000)(aAdpcm,9)
/ CM64
(m10,2000)(aMidi10,10)
(m11,2000)(aMidi10,11)
(m12,2000)(aMidi2,12)
(m13,2000)(aMidi3,13)
(m14,2000)(aMidi4,14)
(m15,2000)(aMidi5,15)
(m16,2000)(aMidi6,16)
(m21,2000)(aMidi11,21)
(m22,2000)(aMidi12,22)
(m23,2000)(aMidi13,23)
(m24,2000)(aMidi14,24)
(m5,2000)(aMidi15,5)
(m6,2000)(aMidi16,6)
(m30,2000)(aMidi10,30)
(m7,2000)(aMidi10,7)
/-----------------------------------------------------------
/ CM64 INIT
.roland_exclusive 16,22=($7F,00,00,00)
/-----------------------------------------------------------
/ ADPCM DATA SET
.adpcm_block_data=FINAL.ZPD
/ ADPCMブロックデータを指定しています
/ ブロックデータを作らない人は上の1行を削除し、
/ 下のADPCMコンフィグレーションのファイルネーム
/ 指定を行っている行の先頭にある '/' を5行分全て
/ 消して下さい。
/ ただしその場合は環境変数 zmusic にADPCM
/ ファイルのあるパスを指定しておくか、もしくは
/ 作業しているディスク上にADPCMファイルが
/ 無ければ鳴りません（詳しくはマニュアル参照）。
/ ADPCMコンフィグレーション
/   1 = flanges_.pcm,v80,p6
/   2 = brts.pcm,v65,p3,m1
/.00c = fck.pcm,v95
/.00d = shps_.pcm,p4,m2,v70
/.00e = clp808.pcm,v130
/-----------------------------------------------------------
/ OPM DATA SET
/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       BASS
(@70,      27, 14,  0,  7,  3, 34,  0,  8,  0,  0,  0
    31, 10,  0,  8,  6, 54,  0,  2,  0,  0,  0
    31, 19,  0,  5,  6, 15,  0,  0,  0,  0,  0
    31,  2,  0,  9,  1,  0,  0,  0,  0,  0,  0
/  AL  FB  SM PAN
    3,  7)
/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       BELL
(@71,      31,  0,  0,  0,  0, 27,  0,  6,  7,  0,  0
    31, 14,  8,  4,  0,  5,  0,  4,  7,  0,  0
```

```
    31,  0,  0,  0,  0, 22,  0,  4,  3,  0,  0
    21, 14,  8,  4,  0,  4,  0,  4,  3,  0,  0
/  AL  FB  SM PAN
     4,  6)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       MAIN
(@72,      31,  0,  0,  0,  0, 37,  0, 14,  7,  0,  0
    21, 13,  9,  4,  2,  3,  0,  4,  7,  0,  0
    31,  0,  0,  0,  0, 33,  0, 12,  3,  0,  0
    21, 13,  9,  4,  2,  3,  0,  4,  3,  0,  0
/  AL  FB  SM PAN
     4,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       BELL2
(@73,      31,  0,  0,  0,  0, 24,  0,  4,  4,  0,  0
    12,  3,  0,  4,  2,  2,  0,  8,  5,  0,  0
    12,  3,  0,  4,  2,  2,  0,  4,  3,  0,  0
    12,  3,  0,  4,  2,  0,  0,  4,  7,  0,  0
/  AL  FB  SM PAN
     6,  6)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       PIANO
(@74,      31,  0,  0,  0,  0, 31,  0,  8,  3,  0,  0
    21, 13,  5,  5,  2,  0,  0,  8,  3,  0,  0
    31,  0,  0,  0,  0, 15,  0,  4,  7,  0,  0
    31, 13,  5,  5,  2,  0,  0,  4,  7,  0,  0
/  AL  FB  SM PAN
     4,  7)

/-------------------------------------------------------------
/ CM64 System SETUP

/ LA SOUND PART
.roland_exclusive 16,22 ={
            $10,0,1                         /address
            0,4,5,                          /reverb
            4,8,4,6,6,0,0,0,4               /ptl reserve
            1,2,3,4,5,6,7,8,9}              /MIDI ch#

/ PCM SOUND PART
.roland_exclusive 16,22 ={
            $52,0,1                         /address
            2,4,4                           /reverb
            4,6,5,5,5,6                     /ptl reserve
            10,11,12,13,14,15}              /MIDI ch#

/-------------------------------------------------------------
/ DRUM SETUP

/ .mt32_drum_setup Note#,devID ={Sound#,level,panpot,revSW}
/ Note# に Sound#（下の音色番号表による）の音をセットします
/ （ MT，CMマニュアル 'Rhythm setup'の項参照 ）

.mt32_drum_setup 36,16 ={64,98, 7,1}          /Bass Drum
.mt32_drum_setup 37,16 ={91,68, 3,1}          /Cabasa
.mt32_drum_setup 38,16 ={65,75, 7,1}          /Snare Drum
.mt32_drum_setup 39,16 ={75,100,7,1}          /Rim Shot
.mt32_drum_setup 41,16 ={68,95, 3,1}          /Low Tom
.mt32_drum_setup 42,16 ={70,60,12,1}          /Closed Hihat
.mt32_drum_setup 44,16 ={93,64,12,1}          /Open Hihat
.mt32_drum_setup 45,16 ={67,91, 7,1}          /Mid Tom
.mt32_drum_setup 46,16 ={86,88, 6,1}          /Tambourine
.mt32_drum_setup 48,16 ={66,87,14,1}          /Hi Tom
.mt32_drum_setup 49,16 ={72,92, 8,1}          /Crash Cym

/ 参考 ： ドラムセットアップにおける
/     音色番号と必要なパーシャル数 （ CM & MT ）

/ Sound Name          No    Ptl#
/ Acou BD   : 64  :   1
/ Acou SD   : 65  :   1
/ Acou Hi Tom         : 66  :   1
/ Acou Mid Tom        : 67  :   1
/ Acou Low Tom        : 68  :   1
/ Elec SD   : 69  :   1
/ Clsd HiHat          : 70  :   1
/ Open HiHat1         : 71  :   2
/ Crash Cym           : 72  :   2
/ Rim Shot  : 73  :   1
/ Hand Clap           : 74  :   1
/ Cowbell   : 75  :   1
/ Mt Hi Conga         : 76  :   1
/ High Conga          : 77  :   1
/ Low Conga           : 78  :   1
/ Hi Timbale          : 79  :   1
/ Low Timbale         : 80  :   1
/ High Bongo          : 81  :   1
/ Low Bongo           : 82  :   1
/ High Agogo          : 83  :   1
/ Low Agogo           : 84  :   1
/ Tambourine          : 85  :   1
/ Claves    : 86  :   1
/ Maracas   : 87  :   1
/ Cabasa    : 91  :   1
/ Quijada   : 92  :   3
/ Open HiHat2         : 93  :   2

/-------------------------------------------------------------
/ MML DATA SET

/   エクスクルーシブデータを送っているので
/   曲が始まる前に少し休符を入れた方が良い。
/   さらに、曲が始まる直前にはふつう音色設定が
/   大量に行われるがその後にも休符をはさむと良い。

(t1)r8
(t2)r8
(t3)r*25
(t4)r*25
(t5)r*25
(t6)r8
(t7)r8
(t9)r8
(t10)r8
(t11)r8
(t12)r*25
(t13)r8
(t14)r*25
(t15)r8
(t16)r*23
(t21)r8
(t22)r8
(t23)r*25
(t24)r8
(t25)r*25
(t26)r8
(t30)r8
(t31)r8

/-------------------------------

/  OPM Bass

(t1)         @k11k-2v15@70p3o2l16q8 t118
(t1)         r2
(t1)         |:q7p3g4.r8q8g8.gr4g4<f&gf&c>g8.gr4<|:q7c4.r8q8c8.cr4:|>
(t1)         q7g4.r8q8g8&<g>gr4g4.r8g8.gr4<q7c4.r8q8c8.cr4c1
(t1)         |:3c8.cr8crc8.cr8<c&d>d8.dr8drd8.dr8g&g-:|
(t1)         e-8.e-r8e-re-8.e-r4f8.fr8frf8.fr4
(t1)         |r*288p2v14(g<e)64,48&(e>f)32,0v15:|>g*576^1g16

/-------------------------------

/  Main Melody (OPM)

/  OPMでソフトLFOを使用している場合、
/  曲の終わりにしばらく休符を入れないと
/  リリースの音にビブラートがかからない。

(t2)         @k0k-2@v0@71p3o3l16q8 @m12 @h33 @s5
(t2)         r4<v14c>rgr
(t2)         |:|:@71d*312r8c8r8f4.r4.|g2.r8gr8bre:|r4^1
(t2)         r^4<@72ar8a8rf+8.r8<d2>r2c4r8b8.rg8.rg8r4.<d8^2
(t2)         r4.>b8r8gl8r.gr.a.re.r.gr2<d+2>r4.<cr2|r*384o2:|
(t2)         o5l16rdr>bl8rdr.>b4^16r1

(t3)         @k0k-2@v0@71p3o3l16q8 @m12 @h33 @s5
(t3)         r4<r>v14bre
(t3)         |:|:r*312d8r8d8r4.e4.|r^2.dr8<c>rgr:|g4^1
(t3)         r^4<r@72b8r8gr8.g8r2d2r4g8r8.ar8.f+r8f+4.<r8^2
(t3)         r4.>r8a8rl8f+.rb.r.gr.f+.rd+2<r2>a4.<rf2|r*360@71o2b:|
(t3)         o5l16grc>rl8grc.>r4^16g*384r1

/  Delay

(t4)         @k-6k-2@v0@71p1o3l16q8 @m12 @h33 @s5 r16.
(t4)         r4<v13c>bge
(t4)         |:|:d*312d8c8d8f4.e4.|g2.rdgr<c>bge:|g4^1
(t4)         r^4<@72p1ab8a8gf+8.g8<d2>d2c4g8b8.ag8.f+g8f+4.<d8^2
(t4)         r4.>b8a8gl8f+.gb.a.ge.f+.gd+2<d+2>a4.<cf2|r*360@71p1o2b:|
(t4)         o5l16gdc>bl8gdc.>b4^16g*384r16

(t25)        @k7k-2@v0@71p2o3l16q8 @m12 @h33 @s5 r8
(t25)        r4<v13c>bge
(t25)        |:|:d*312d8c8d8f4.e4.|g2.rdgr<c>bge:|g4^1
(t25)        r^4<@72p2ab8a8gf+8.g8<d2>d2c4g8b8.ag8.f+g8f+4.<d8^2
(t25)        r4.>b8a8gl8f+.gb.a.ge.f+.gd+2<d+2>a4.<cf2|r*360@71p2o2b:|
(t25)        o5l16gdc>bl8gdc.>b4^16g*384r1

/-------------------------------

/  Sub Melody & Piano

(t26)        @k-6k0v10@73p3o2l16q7 @m3 @h10 @s6
(t26)        r2
(t26)        |:@73o2p3a4.<r1a8r8.a8.r8d4.<r*384
(t26)        c2r8d8.r8.>a8^4.r16a16r4.<d8^1
(t26)        l16rc>rfrc>r8@74o2p1
(t26)        d8.dr4d8ddr4e8.er4e8eer4|:@74p1d8.dr4d8ddr4e8.er4e4e4:|
(t26)        |:f8.f^4:||:g8.g^4:|
(t26)        |r*384:|a*576p3a16r1

(t31)        @k6k0v10@73p3o2l16q7 @m3 @h10 @s6
(t31)        r2
(t31)        |:r4.@73p3o2<a1r8b-8.r8.f8r4.<d*384
(t31)        r2d8r8.c8.>r8^4.g16r16f4.<r8^1
(t31)        l16dr>b-rdr>b-r@74o1p2
(t31)        b-8.b-r4b-8b-b-r4<c8.cr4c8ccr4|:@74p2>b-8.b-r4b-8b-b-r4<c8.cr4c4c4:|
(t31)        |:c+8.c+^4:||:d+8.d+^4:|
(t31)        |r*384:|f*576p3f16r1

/-------------------------------

/  ADPCM Rhythm

(t9)         l4o0@d1
(t9)         r2
(t9)         |:|:|:3cdc8.c16d:||cdc8.c16d16d16r8:|crc8.c8.r8
(t9)         |:3|:c8.c16dcd8.|r16:|c16:||:c8.c16dcd8.|c16:|r16
(t9)         |l16cr8cdr|:cr8:|dr8d rcccdrcrcr8cr*2e*46:|

```

```
/--------------------------------
/  Main Melody ( LA & PCM )
(t13)       k-2 @v105 @u100 @98 p3 o4 l16 q8@k3@m30
(t13)       r4<c>bge
(t13)       |:|:d*312d8c8d8f4.e4.|g2.rdgr<c>bge:|g4^1
(t13)       r4r<ab8a8gf+8.g8'b2<d'd2c4g8b8.ag8.f+g8'a4.f+'<d8^2
(t13)       r4.>b8a8gl8f+.g b.'f+8.a''e8g''c8.e''d8.f+''e8g'
(t13)       '>b-2<d+''b-2<d+''f4.a'<cf2|r*360o3b:|
(t13)       o6l16gdc>bl8gdc.>b4^16g*384
(t22)       k-2 @v106 @u100 @9 p3 o4 l16 q8 @K-5
(t22)       r4<c>bge
(t22)       |:|:@u100d*312d8c8d8f4.e4.|g2.rdgr<c>bge:|g4^1
(t22)       @u95r4r<ab8a8gf+8.g8b2<d2 >c4g8b8.ag8.f+g8f+4.<d8^2
(t22)       r4.>b8a8gl8f+.g b.'f+8.a' 'e8g' 'c8.e' 'd8.f+' 'e8g'
(t22)       '>b-2<d+''b-2<d+''f4.a'<cf2|r*360o3b:|
(t22)       o6@u90l16gdc>bl8gdc.>b4^16g*384
/--------------------------------
/  Stringth ( LA & PCM )
(t14)       k-2 @v60 @u76 @51 @p121o5 l1 q8@m60@h30
(t14)       r2
(t14)       |:@u76b*384g*384b*288<d2>g*360a8
(t14)       @u78baba2g4g-4bal8g4aa+<c.d.d+>a4a+<cd4d+4
(t14)       |r*384:|'dg'576
(t5)        k-2 @v64 @u76 @36 @p55 o5 l1 q8@k1
(t5)        r2
(t5)        |:@u76'gb'384'eg'384'gb'288'b<d'96'eg'360a8
(t5)        @u78baba2g4g-4bal8g4aa+<c.d.d+>a4a+<cd4d+4
(t5)        |r*384:|'dg'576
/--------------------------------
/  Sub Melody & Piano (LA)
(t15)       k0 @v99 @u66 @034 @p68 o4 q8
(t15)       r2
(t15)       |:@u55@34p3r4.< 'c1fa' 'c8fa' 'c8.gb-''c8.fa''c8f''>b-4.<d''fb-<d'384
(t15)       'f2a<c'<d8d8.c8.>a8^4.'e16g''f16a''>b-4.<df''b-f'<d'216
(t15)       l16<dc>b-fdc>b-f>
(t15)       @3@u64@p100'f8.b-<d''fb-<d'r4'f8b-<d''fb-<d''fb-<d'r4
(t15)       'g8.<ce''g<ce'r4'g8<ce''g<ce''g<ce'r4
(t15)       |:'f8.b-<d''fb-<d'r4'f8b-<d''fb-<d''fb-<d'r4'g8.<ce''g<ce'r4'g4<ce''g4<ce':|
(t15)       |:'g+8.<c+f''g+<c+f'60:||:'b-8.<d+g''b-<d+g'60:|
(t15)       |r*384:|'<cfa'576@u74'fa<cf'@v0
(t16)       k0 @v99 @u79 @100 @p24 o4 q7 @k-1
(t16)       r2
(t16)       |:@u75@7@p52r4.< 'c1fa' 'c8fa' 'c8.gb-''c8.fa''c8f''>b-4.<d''fb-<d'384
(t16)       'f2a<c'<d8d8.c8.>a8^4.'e16g''f16a''>b-4.<df''b-f<d'216
(t16)       l16<dc>b-fdc>b-f>
(t16)       @3@u60@p28'f8.b-<d''fb-<d'r4'f8b-<d''fb-<d''fb-<d'r4
(t16)       'g8.<ce''g<ce'r4'g8<ce''g<ce''g<ce'r4
(t16)       |:'f8.b-<d''fb-<d'r4'f8b-<d''fb-<d''fb-<d'r4'g8.<ce''g<ce'r4'g4<ce''g4<ce':|
(t16)       |:'g+8.<c+f''g+<c+f'60:||:'b-8.<d+g''b-<d+g'60:|
(t16)       |r*384:|'<cfa'576@u74'fa<cf'@v0
/--------------------------------
/  PCM Bass
(t21)       k-2 v13 @u104 @23 p3 o1 l16 q8
(t21)       x$50,0,8,0 r2
(t21)       |:q7g4.r8q8g8.gr4g4v14<@u108f@u80g@u105f@u80c>v13@u105g8.gr4
(t21)       <|:q7c4.r8q8c8.cr4:|>
(t21)       q7g4.r8q8g8<@u100g>@u101gr4g4.r8g8.gr4 <q7c4.r8q8c8.cr4c1
(t21)       |:3c8.q7c8rq8crc8.q7c8rv15<q8@u108c@u70d@u105>
(t21)       v13d8.q7d8rd8q8d8.q7d8rq8@u108g@u90g-@u105:|
(t21)       e-8.q7e-8rq6e-8q8e-8.e-r4f8.q6f8rf8q8f8.fr4
(t21)       |r*288@v80g2@v105:|>g*576@v110@u107g16
/--------------------------------
/  Electric Organ Seq.
(t12)       k-2 @v55  @u100 @21 @p107o6 l16 q2
(t12)       r2
(t12)       |:38r'dg'a'dg'a'dg'dg a'dg'aaraag:| ¥35
(t12)       |:4 r'dg'a'dg'a'dg'dg a'dg'aaraag:|
(t23)       k-2 @v55  @u100 @44 @p20 o6 l16 q2 @k2
(t23)       r2
(t23)       |:38g'd16g'r'd16g'a'd16a'dr a'd16g'argaar:| ¥35
(t23)       |:4 g'd16g'r'd16g'a'd16a'dr a'd16g'argaar:|
/--------------------------------
/  Sax
(t24)       k0 @v73 @u106 @57 @p78 o3 l8 q8 @k-2 @m120@h40
(t24)       r2
(t24)       |:@u110|:f2>f8.f16r2.f8.f16r<c8>'a+f'384:|<
(t24)       @u100|:3@p53'd8.b-''d16b-'r2.@p78'>g8.<e<c''>g16<e<c'r2.:|f1g1|r*384:|'ca'576
/--------------------------------
/  Guiter
(t6)        k-2 @v80 @u101 @10 @p30 o5 l8 q8
(t6)        r2@v0@u0
(t6)        |:q8@p44@10o5|:r4.d>a.g.d^1r4.ge.d.>g1|r:|
(t6)        @12@v73@u105@p30q8r<
(t6)        |:'c.e''c16e'r'>g<c''ce'|:'c16e':|r4'd.f+''d16f+'r16|:3'df+16':||:'d4f+':|
(t6)        |'c.e''c16e'r'>g<c''ce'|:'c16e':|r4'd.f+''d16f+'r16|:3'df+16':|'d4g''d4f+':|
(t6)        'd+.g''d+16g'r16|:3'd+16g':||:'d+4g':|
(t6)        'f.a''f16a'r16|:3'f16a':|'f.a''fa'60
(t6)        r*384@v110@u106:|
/--------------------------------
/  LA Rhythm
(t10)       @u123 q8 o2 l16
(t10)       @v100<c64@v115c32.>r8aaafr
(t10)       |:|:r1 c+r8c+r2r8|r8:|af|:r1|c+r8c+r2.:|
(t10)       c8@v95<c8c@v100c>@v102a8@v113<cc>r@v115aarff
(t10)       |:|:4c+r8c+r8c+r^2:|:||c+<cc>c+r8'c+16a'arfr4.
(t10)       c+r8c+r8c+r<c>afr^4:||:3c+r8c+r2.:|
(t11)       l4o2
(t11)       r2
(t11)       |:|:|:3cdc8.c16d:||cdc8.c16d16d16r8:|crc8.c8.r8
(t11)       |:3c8.c16dcdc8.c16dcd8.c16:|c8.c16dcd8.c16c8.c16dcd
(t11)       |l16cr8cdrcr8cr8dr8d ccccdrcrcr8cr4:|
(t30)       @u127l8 q8 @i65,16,22
(t30)       o2 r2
(t30)       |:|:7rf+rf+rf+r'a+f+':|r1|:4rf+:|
(t30)       |:7@u127rg+rf+rf+rg+:|@u127|rg+r4.g+rg+rg+r4.f+
(t30)       @v127 x$10,0,2,7,7 d+@v115f+ x$10,0,2,4,5:||:12rf+:|
(t7)        @u116 l1 q8 @d1
(t7)        o3 r2
(t7)        |:c+r*576c+r*384c+|:c+r*576:||o7@u76a+*384o2@u116:|c+
(p)
```

リスト2

2-a

```
===============  KEEP.CNF  ==================
1=ctrlk.pcm,v40
2=gatek.pcm,v65,m1
3=flangs.pcm,p-1,v40
4=gate_sd.pcm,v30,m3
.O0c =dsb.pcm,v40,m2
.O0d =rvbs1_.pcm,v110,m4
.O0d+=14,v75,mO0c
.O0e =14,v75
.O0f =14,v50
.O0a =crsh1.pcm,v140,mO0c
.O0b =flanges_.pcm,v120
.O1c =tom5.pcm,v130
.O1c+=24,mO0c
.O1d =tom6.pcm,v130
.O1d+=26,mO0c
.O1e =tom7.pcm,v130
.O1e+=28,mO0c
.erase 1
.erase 2
.erase 3
.erase 4
```

2-b

```
===============  KEEP.ZMS  ==================
.comment -TURBO OUTRUN- Keep Your Heart (C)SEGA  Programmed by ENG
/ 1991-07-29
/ for ZMUSIC.X
/--------------------------------------------------------------
/ TRACK SETUP
(i)
(b0)                          / Base Channel = Internal
(m1,2000)(aFm1,1)
(m2,2000)(aFm2,2)
(m3,2000)(aFm3,3)
(m4,2000)(aFm4,4)
(m5,2000)(aFm5,5)
(m6,2000)(aFm6,6)
(m7,2000)(aFm7,7)
(m8,2000)(aFm8,8)
(m9,2000)(aAdpcm,9)
```

▶ZMUSIC.X広告の題は「ウッキーゼンジとその仲間たち」でしょうか？
佐藤　正明(18)福岡県

```
/
/-----------------------------------------------------------
/ ADPCM DATA SET

.ADPCM_BLOCK_DATA=KEEP.ZPD

/-----------------------------------------------------------
/ OPM DATA SET

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       BASS 1
(@70,      31,  8,  3,  8,  2, 26,  3,  6,  0,  0,  0
     24, 12,  3,  8,  4, 47,  2, 13,  7,  0,  0
     31, 10,  3,  8,  2, 14,  2,  0,  3,  0,  0
     31,  4,  3,  7,  2,  1,  2,  1,  3,  0,  0
/  AL  FB  SM PAN
     0,  5)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       HIHAT
(@71,      31,  0,  0,  5, 12, 10,  0, 15,  3,  1,  0
     31,  0,  0,  5, 10, 23,  0, 10,  3,  3,  0
     31,  0,  0,  5, 10, 13,  0, 15,  0,  3,  0
     23, 15, 14,  8,  9,  8,  0,  1,  6,  0,  0
/  AL  FB  SM PAN
     3,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       RIDE
(@72,      31,  0,  0,  0,  0,  0,  0, 10,  0,  1,  0
     20, 31, 11,  5,  1,  0,  0, 11,  0,  2,  0
     31, 28,  5,  3,  3, 17,  0,  1,  0,  2,  0
     20, 31, 11,  5,  4,  0,  0, 14,  0,  3,  0
/  AL  FB  SM PAN
     4,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       CRASH
(@73,      31, 31,  0,  0,  0,  0,  0, 11,  0,  3,  0
     31, 31,  0,  0,  1,  0,  0, 14,  0,  3,  0
     31, 22,  0,  0,  1,  0,  0, 15,  7,  1,  0
     31, 18,  7,  4,  0,  0,  2, 15,  0,  0,  0
/  AL  FB  SM PAN
     1,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SHAKER
(@74,      31, 31,  0,  3,  1,  2,  0, 11,  0,  3,  0
     31, 28,  0,  3,  1,  0,  1, 12,  0,  3,  0
     31, 22,  0,  3,  1,  5,  0,  1,  7,  1,  0
     18, 14, 10,  9,  0,  0,  1,  7,  0,  0,  0
/  AL  FB  SM PAN
     1,  5)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SYNTH1
(@75,      31,  0,  0, 15,  0, 24,  0,  8,  3,  0,  0
     31,  0,  0, 15,  0,  2,  0,  8,  7,  0,  0
     31,  0,  0, 15,  0, 23,  0,  4,  7,  0,  0
     31,  0,  0, 15,  0,  2,  0,  4,  3,  0,  0
/  AL  FB  SM PAN
     4,  5)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SYNTH2
(@76,      31, 18,  0, 10, 15, 11,  0,  7,  5,  0,  0
     31, 13, 18,  7,  1,  1,  1,  0,  3,  0,  0
     31, 13, 18,  7,  1,  1,  1,  1,  3,  0,  0
     31, 13, 18,  7,  1,  1,  1,  1,  7,  0,  0
/  AL  FB  SM PAN
     5,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SYNTH3
(@77,      31, 14,  0,  0,  1, 22,  0,  2,  0,  0,  0
     31,  0,  0,  0,  0, 29,  0,  2,  0,  0,  0
     31,  0,  0,  0,  0, 26,  0,  2,  0,  0,  0
     31,  0,  0,  5,  0,  0,  0,  2,  1,  0,  0
/  AL  FB  SM PAN
     3,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SYNTH4
(@78,      31,  0,  0,  4,  0, 26,  0,  8,  3,  0,  0
     31, 14,  0,  6,  1,  1,  0,  8,  3,  0,  0
     31,  0,  0,  4,  0, 25,  0,  4,  7,  0,  0
     31, 14,  0,  6,  1,  1,  0,  4,  7,  0,  0
/  AL  FB  SM PAN
     4,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       SUB
(@79,      21,  1,  1,  1,  0, 30,  1,  2,  0,  0,  0
     21,  1,  1,  1,  0, 27,  1,  2,  1,  0,  0
     21,  1,  1,  4,  0, 40,  1, 10,  0,  0,  0
     24,  1,  1,  8,  1,  0,  1,  2,  1,  0,  0
/  AL  FB  SM PAN
     0,  5)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       BASS 2
(@80,      31,  7,  3,  6,  2, 27,  3,  6,  0,  0,  0
     24,  6,  3,  3,  1, 56,  3,  9,  7,  0,  0
     31,  9,  3,  2,  1, 18,  2,  0,  2,  0,  0
     31,  4,  3,  7,  2,  0,  2,  1,  3,  0,  0
/  AL  FB  SM PAN
     0,  4)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       CHORD
(@81,      31,  0,  0,  0,  0, 30,  0,  6,  3,  0,  0
     31,  0,  0,  5,  0,  0,  0,  6,  7,  0,  0
     31,  0,  0,  0,  0, 22,  0,  4,  7,  0,  0
     31,  0,  0,  5,  0,  0,  0,  4,  3,  0,  0
/  AL  FB  SM PAN
     4,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       D.G H
(@82,      27, 25,  1,  9,  2, 27,  0,  4,  3,  2,  0
     27, 20,  1,  9,  4, 24,  1, 14,  6,  0,  0
     31,  2,  1,  9,  3,  7,  0,  0,  3,  0,  0
     30, 14,  0,  9,  0,  1,  1,  1,  7,  0,  0
/  AL  FB  SM PAN
     3,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       D.G L
(@83,      27, 17,  1,  8,  3, 26,  0,  3,  7,  1,  0
     27, 16,  1,  8,  4, 23,  0, 12,  3,  0,  0
     31,  2,  1,  8,  3, 10,  0,  0,  7,  0,  0
     31,  0,  0,  8,  0,  0,  1,  1,  3,  0,  0
/  AL  FB  SM PAN
     3,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       D.G M
(@84,      19, 16, 17,  8,  9, 16,  0,  3,  7,  1,  0
     31, 16, 17,  8,  4, 27,  1, 12,  3,  0,  0
     31, 16, 18,  8,  3, 14,  0,  0,  7,  0,  0
     22, 14, 10, 11,  1,  0,  1,  1,  3,  0,  0
/  AL  FB  SM PAN
     3,  7)

/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME       MAIN
(@85,      16, 15,  0,  8,  3, 22,  1,  3,  0,  0,  0
     19,  0,  0,  7,  3, 26,  0,  1,  3,  0,  0
     26,  0,  0,  7,  3, 26,  0,  1,  3,  0,  0
     27,  0,  0,  7,  3,  3,  1,  1,  3,  0,  0
/  AL  FB  SM PAN
     0,  7)

/-----------------------------------------------------------
/ MML DATA SET

/  Electric Bass
/  `=[SHIFT]+[@]

(t1)         r4[do]@70o2v15@q3p318@k0
(t1)         |:4a&|:29a:|g4:|
(t1)         q8g*384a1^4re2.e<c>bq6a&
(t1)         |:|:4|:15a:|g&|:15g:||a&:|<d&
(t1)         |:4|:7d:|>a&|:7a:|g&|:13g:||a4<d&:|rdq8d*312q6c+
>ba|a&:|d+
(t1)         @80o2q8|:8|:3|:5e:|e4e&:|`d4.c+4d4d+:|
(t1)         v16|:4q8ddr4q7c+4q8ddr4q7c+4.|1e4.:||2e4.:||3>a4
.<:||4q8e*264&:|
(t1)         |:5e&:|e&e@70o3v15q618d&
(t1)         |:8|:7d:|>a&|:7a:|g&|:13g:||a4<d&:|rdq8d*312q6c+
>ba

/------------------------------

/  Distortion Guitar (lower)

(t2)         r*49[do]@83o3v14q8p318@k6d&
(t2)         |:4d4d4dd4d4.d4dd4>a1&a2&a4g8..&v9g32<v14|d&:|
(t2)         q8>g*384a1^4rv15e2&e&q7ev14e<c>bv15
(t2)         |:|:4@83o2v14a4&v13a2.&a1v14g4&v13g2.&g1:|
(t2)         v14q8|:3<d1>a1g*336a4:|<d1>a1g*336r<dd&d1&|d2..:
|
(t2)         d2..&d+@83o3v14q718~2
(t2)         |:8|:3|:5e:|e&ee&:|`d4&dc+&c+d&d&d+:|
(t2)         |:|:q8v14ddv9dv7dq7v14q8|1c+8..&v9c+32:||2c+4&v9
c+8:|v14e4^16.&v9e32:|
(t2)         |:|:q8v14ddv9dv7dq7v14q8|1c+8..&v9c+32:||2c+4&v9
c+8:|
(t2)         |1>v14a4^16.&v9a32<:||2>q8v14e4&e1&e1:|
(t2)         v14q8|:8<d1>a1g*336|a4:|r<dd*384

/------------------------------

/  Distortion Guiter (upper)

(t3)         r4[do]@82o4v14q8p318@k-3d&
(t3)         |:4q7d4q8dv9dv14ddv9dv14d4v9dv14dv9dv14ddv9d>v14
q7a1&a2&a4q8g8..&v9g32<v14|d&:|
(t3)         q8v15>g*384a1^4re^2&q7ee<c>bv14
(t3)         |:|:4@82o3q6a4@84q5|:14a:|@82q6g4@84q5|:14g:|@82
:|
(t3)         v14q8|:3<d1>a1g*336a4:|<d1>a1g*336r<dd&d1&|d2..:
|
(t3)         d2..&d+@82o4v13q718
(t3)         |:8|:3|:5e:|e&ee&:|`d4&dc+&c+d&d&d+:|v14
(t3)         |:|:q8v14ddv9dv7dq7v14q8|1c+8..&v9c+32:||2c+4&v9
c+8:|v14e4^16.&v9e32:|
(t3)         |:|:q8v14ddv9dv7dq7v14q8|1c+8..&v9c+32:||2c+4&v9
c+8:|
(t3)         |1>v14a4^16.&v9a32<:||2>q8v14e*432:|
(t3)         v14q8|:8<d1>a1g*336|a4:|r<dd*384

/------------------------------

/  Chord & Guit.

(t4)         r4[do]@82o4v11q7p118@k8
(t4)         @h42@s6@m
(t4)         |:4p1o4v10q7d4.q8dv8p2dv10p1ddv8p2dv10p1d4v8p2dv
10p1dv8p2dv10p1ddv8p2d
(t4)         v10>p1q7a*288&a4q8g4<:|r@76p1
(t4)         o3q5v13|:5gf+d:||:3ag+e:|ag+18@77o3v8p1q7e2..r4
(t4)         |:|:4@78q8o3q6v12p3c+4v10p1r4v12p3d4v10p1r4.>v12
p3b4<v10p1rv12p3c+4v10p1r4.
(t4)         >|:v12p3b4v10p1rv12p3b4v10p1r4|r:|:|
(t4)         @m23@77q8o3v13p1|:4a1a1g*336|g8&v9g8v13:|raar*36
0:|
(t4)         r@76p1@m
(t4)         o2q6v13|:16bb<e4>b<d>bb<e>bb<@k-2a-.>@K8b16b<a->
b:|<|:26dgf+:|d
(t4)         @m23@77q8o3v13p1|:8a1a1g*336|g8&v9g8v13:|raar*36
0

/------------------------------

/  Chord & Sub
```

▶西川善司さんへ。シンナー野郎はアンパンマンですが，薬（ヤク）づけ野郎はシャブオジサンというそうです。
秋友　謙二(18)山口県

```
(t5)          r*49[do]q8l8@k0
(t5)          @h42@s6@m
(t5)          |:4@81o3v10d*384@77o4v11p2e<d&>a&g<c+&>a&g<d&>a&g<c+&>a&gb&v7g&d:|r@76p2@l2rl8
(t5)          o3q5v13|:5gf+d:||:3ag+e:|a@l22g+l8@77o3v8p2q7b2..r4
(t5)          |:|:4@78q8o2q6v12p3a4v10p2r4v12p3a4v10p2r4.v12p3g4v10p2rv12p3a4v10p2r4.
(t5)          |:v12p3g4v10p2rv12p3g4v10p2r4|r:|:|
(t5)          @m23@77q8o3@v120f+2&f+8l8@v118<def+>@v120e2&e8l8@v118<d4c+>@v120d2.&d1l8e8&v9e8
(t5)          |:@v120f+2&f+8def+e2&e8l8@v118<d4c+>@v120d2.&d1l8e8&v9e8:|
(t5)          @77q8o3@v120f+2&f+8l8@v118<def+>@v120e2&e8l8@v118<d4c+>@v120d2.&d1l8r<ddr*360:|
(t5)          r@76p2@l2rl8@m
(t5)          o2q6v13|:16bb<e4>b<d>bb<e>bb<@k-11a-.>@k0b16b<a->b:|<|:26dgf+:|@l22d
(t5)          @m23@77q8
(t5)          |:o3@v120f+2&f+8l8@v118<def+>@v120e2&e8l8@v118<d4c+>@v120d2.&d1l8e8&v9e8
(t5)          |:@v120f+2&f+8l8def+e2&e8l8@v118<d4c+>@v120d2.&d1l8e8&v9e8:|
(t5)          o3@v120f+2&f+8l8@v118<def+>@v120e2&e8l8@v118<d4c+>@v120d2.&d1l8|e8&v9e8:|r<ddr*360
/--------------------------------
/  Main Melody 1
/ トラック6，7共に (B),(D),(E) はほとんど同じです
/  (A)
(t6)          r4[do]@85o4v0q8p3l8@k4@h50@s7@m
(t6)          |:4r4v13p3ev8p2ev6p1ev13p3ev8p2ev6p1ev3p2ev13p3ev8p2ev6p1ev13p3ev8p1ev13p3ev8p2e<v13p3e
(t6)          <@v124d&>v14a&g<@v124c+&>v14a&g<@v124d&>v14a&g<@v124c+&>v14a&g@v123b&v13g&v10d>>:|
(t6)          @m9@77o3v13(g,a)384,360a1^4v9rv14q7e*168r4@m
(t6)          |:@85o3v15q8p1l8
(t6)          (b32<c),0&c+*306>a4
(t6)          ab&v10bv15<c+>@b40,-340,220b1^4@b{a&v10a}{v15e&v10e}v15q8a<c+1e2
(t6)          (g16,a)&a16^2a4.e4.g2.&(g4,f)&v10fv15
(t6)          (a,b)1l&b*13&a*23&e*23a*2&|:5b8&|a&e:|e
(t6)          (b<c+)24&c+8ec+&v10c+v15>a4b2.a&v10av15
(t6)          f+2e1{dc+}dc+>ba&a2.<c+4>(a8,b)&b4a4g&v10g
/  (B)
(t6)          @79o3v16l8q8@m23
(t6)          (e8,f+)&|:f+2def+e^2d4c+
(t6)          |1d^1(g8,a)&ag4f+4{e&p1v10ep3}@v126f+&:|
(t6)          |2>b*264l16f+gab<c+&p2v11c+p3@v126d&p1v11c+p3@v126e&p2v11dp3@v126l8:|
(t6)          |:f+^2def+e^2
(t6)          |1d4c+d*312ag&p1v11gp3@v126:|
(t6)          |2(c8,d)&dc+>b*336&v12b@v126<<@m
(t6)          (c+32,d)&d16.d&p2v9d&p1v8d&p2v7d&p1v6d&p2v5d&p1v4d&p2v3d&p1v2d&p2v1dr2.@v98:|:|
/  (C)
(t6)          r|:@75o2q8b*1536&@85o3v15q8p1
(t6)          q8|:(d16,e)&e4..b4<{c+&v10c+v15}(c+4,d)&
(t6)          |1d2..d2c+4.>b4<{c+&v10c+v15}>e2&v10e&ev15:|
(t6)          |2<d2.c+d2c+2>b&(b4,a-)<e&e2@v112:|:|
(t6)          <v15p2|:4v15ddv8dv7dv6dv5dv15ddv8dv7dv6dv5dv4dv3dv2dv1d:|@71o4v14p1q1|:7gr:|g
/  (D)
(t6)          @79o3v16l8q8@m23
(t6)          (e8,f+)&|:f+2def+e&e2d4c+
(t6)          |1d^1(g8,a)&ag4f+4{e&p1v10ep3}@v126f+&:|
(t6)          |2>b*264l16f+gab<c+&p2v11c+p3@v126d&p1v11c+p3@v126e&p2v11dp3@v126l8:|
(t6)          |:f+^2def+e^2
(t6)          |1d4c+d*312ag&p1v11gp3@v126:|
(t6)          |2(c8,d)&dc+>b*336&v12b4o4@v126:|
/  (E)
(t6)          (e8,f+)&|:f+2def+e&e2d4c+
(t6)          |1d^1(g8,a)&ag4f+4{e&p1v10ep3}@v126f+&:|
(t6)          |2>b*264l16f+gab<c+&p2v11c+p3@v126d&p1v11c+p3@v126e&p2v11dp3@v126l8:|
(t6)          |:f+^2def+e^2
(t6)          |1d4c+d*312ag&p1v11gp3@v126:|
(t6)          |2(c8,d)&dc+>b*336&v12b@v126<<@m
(t6)          (c+32,d)&d16.d&p2v9d&p1v8d&p2v7d&p1v6d&p2v5d&p1v4d&p2v3d&p1v2d&p2v1dr2.@v98:|
/  Main Melody 2
/  (A)
(t7)          r4[do]@85o3v0q8p3l8@k-4@h50@s7@m
(t7)          |:4r4v16p3av9p1av7p2av16p3g&v14av9p2av7p1av16p3av9p1av7p2a
(t7)          <v16p3c+v9p2c+v16p3dv9p1dv7p2d
(t7)          v5p1d@v125>p3av8p2av6p1a@v125p3g&v14av8p1av6p2a
(t7)          @v125p3av8p2av6p1a@v125p3gv8p1gv6p2gv4p1g:|
(t7)          @m9@77o3v13(b<c+)384,360c+1^4v11rv14q7>a-*168r4@m
(t7)          |:@85o3v15q8p2l8
(t7)          (b32<c),0&c+*306>a4
(t7)          ab&v10bv15<c+>@b-40,-260,220b1^4@b{a&v10a}{v15e&v10e}v15q8a<c+1e2
(t7)          (g16,a)&a16&a2a4.e4.g2.&(g4,f)&v10fv15
(t7)          (a,b)1l&b*13&a*23&e*23a*2&|:5b8&|a&e:|e
(t7)          (b<c+)24&c+8ec+&v10c+v15>a4b2.a&v10av15
(t7)          f+2e1{dc+}dc+>ba&a2.<c+4>(a8,b)&b4a4g&v10g
/  (B)
(t7)          @79o3@v124l8q8@m23
(t7)          (c8,d)&|:d2>b<c+dc+^2>b4a<
(t7)          |1>b^1<(d+8,f+)&f+e4d4{c+&p2v9c+}p3@v123d&:|
(t7)          |2>g*264l16def+ga&p1v10ap3@v123b&p2v10ap3@v123<c+&p1v10>bl8p3@v123<:|
(t7)          |:d^2>b<c+dc+^2
(t7)          |1>b4ab*312<f+e&p2v10e@v123p3:|
(t7)          |2>(b-8,b)&bag*336&v12g@m
(t7)          @v123<aa&p1v9a&p2v8a&p1v7a&p2v6a&p1v5a&p2v4a&p1v3a&p2v2a&p1v1ar2.@v88:|:|
/  (C)
(t7)          r|:@75o2v5q8g+*1536&@85o3v15q8p2
(t7)          q8|:(d16,e)&e4..b4<{c+&v10c+v15}(c+4,d)&
(t7)          |1d2..d2c+4.>b4<{c+&v10c+v15}>e2&v10e&ev15:|
(t7)          |2<d2.c+d2c+2>b&(b4,a-)<e&e2@v98:|:|
(t7)          v15p1|:4v15aav8av7av6av5av15aav8av7av6av5av4av3av2av1a:|@71o4v13p3q1|:7gr:|g
/  (D)
(t7)          @79o3@v124l8q8@m23
(t7)          (c8,d)&|:d2>b<c+dc+^2>b4a<
(t7)          |1>b^1<(d+8,f+)&f+e4d4{c+&p2v9c+}p3@v123d&:|
(t7)          |2>g*264l16def+ga&p1v10ap3@v123b&p2v10ap3@v123<c+&p1v10>bl8p3@v123<:|
(t7)          |:d^2>b<c+dc+^2
(t7)          |1>b4ab*312<f+e&p2v10e@v123p3:|
(t7)          |2>(b-8,b)&bag*336&v12g4o4:|
/  (E)
(t7)          (c8,d)&|:d2>b<c+dc+^2>b4a<
(t7)          |1>b^1<(d+8,f+)&f+e4d4{c+&p2v9c+}p3@v123d&:|
(t7)          |2>g*264l16def+ga&p1v10ap3@v123b&p2v10ap3@v123<c+&p1v10>bl8p3@v123<:|
(t7)          |:d^2>b<c+dc+^2
(t7)          |1>b4ab*312<f+e&p2v10e@v123p3:|
(t7)          |2>(b-8,b)&bag*336&v12g@m
(t7)          @v123<aa&p1v9a&p2v8a&p1v7a&p2v6a&p1v5a&p2v4a&p1v3a&p2v2a&p1v1ar2.@v88:|
/--------------------------------
/  OPM Rhythm (hihat) & Chord
(t8)          @h42@s6@m018
(t8)          t176rt190r@71o4v16q1l8@k11@ort169[do]
(t8)          |:4|:27g:|{gg}@73ap1cr|r@71:|
(t8)          r@72p1|:5cr:|{c&c}rcrcr
(t8)          r*432r@71g|:
(t8)          |:3|:4ggg|q1g:|q1{gg}
(t8)          |:3ggg|q1g:|q1{gg}gg@o21g@oq1@72p1g@71:|
(t8)          |:4ggg|q1g:|q1{gg}
(t8)          |:gggq1g:|ggg{gg}
(t8)          @74q4p2v16|:4g16 _4:|v16r@ml8@77q8o4p2v12d&
(t8)          |:4d2..c+1>b*336<|c+8&v9c+8v12d&:|raar4
(t8)          r2@m@71o4v16q1{gg}q7g4q1grgrgr|g:|
(t8)          @72@v124p1c@71o4l8v16q1|:8@o10p3g@op1g:|
(t8)          |:6@o10p3g@op1g:|{@o10p3gg@o}@73p1c@71
(t8)          |:17@o10p3g@op1g:|
(t8)          |:64v15p1q8g_3p2q1g:|
(t8)          |:4@74q4p2v16|:4g16 _4:|@71|:7v16p1g_4p2g:|:|
(t8)          @75o2q8v9@b0,7200,1752b*1872r8@b0@ml8@77o4p2v12d&
(t8)          |:8d2..c+1>b*336<|c+8&v9c+8v12d&:|raar*384@mv16@71o4q1
/--------------------------------
/  ADPCM Rhythm
(t9)          l8o0
(t9)          {dd}da4[do]
(t9)          |:4cd4|:5ccd4:|ccd{dd}c<d{e+e}>|a4:|
(t9)          cc4c4<e+4>c4c4<{c+c}d>c<d>cc4cdcccd<{c+d}e+4>a2.cddc4
(t9)          |:|:4cd4|:ccd4:|ccd8.c16|:ccd4:|c|cd{de}<e+>cdc4:|
(t9)          <d+d>c{<d+d}e+>da4
(t9)          |:cd4ccdc4cd4.cdc4cd4ccd|1c4cdcccda4:||2d+{d+d<p1ccp3d+dp2ee>}2p3dcac4:|
(t9)          |:cd4ccdc4cd4.cdc4cd4ccd |c4cdc4cda4:|
(t9)          ccc{d<p1cp3ddp2ee>}4p3d4@f3a@f4a2
(t9)          {cc}d<e4>b4<d4de>cd|c4:|a
(t9)          |:7c4:|c{ef}|:6c4:|c16c.c{ef}
(t9)          c4<d+16d.>c<d>c4c<e>c<{ee>}{<c+d}e>c{ef}
(t9)          c4<e+4>c<e>c4<d+4>c8{d+f}<d+de+>a
(t9)          |:|:3c4d+4:|c4cec4d+4c4cec4d+4|<{c+d}e>d+a:|c<e>d+e
(t9)          y2,21|:3|:4r4|d+4y2,12:|d+dy2,12:|
(t9)          r4d+4cec{ee}c<dd+>c<{c+d}e>ad
(t9)          |:3|:c4d+4cc|d+4:|<e+>e:|
(t9)          c4c4ced+4c<de+>ed+e<e+>a
(t9)          |:4cc4.d4cc4.|1d4.a4.:||2d4{<c+cddee>}2:||3d4.a4.:||4d4.a4&a1&addddd<de>a4:|
(t9)          |:3|:cd4ccdc4cd4.cdc4cd4ccd|1c4cdcccda4:||2d+{d+d<p1ccp3d+dp2ee>}2p3dcac4:|:|
(t9)          |:cd4ccdc4cd4.cdc4cd4ccd|c4cdc4cda4:|ccc{d<p1cp3ddp2ee>}4p3d4@f3a@f4a2
(t9)          {cc}d<e4>b4<d4de>cda4
/--------------------------------
(t1)          [loop]
(t2)          [loop]
(t3)          [loop]
(t4)          [loop]
(t5)          [loop]
(t6)          [loop]
(t7)          [loop]
(t8)          [loop]
(t9)          [loop]
(p)
/   繰り返し記号を多く使ったため曲の長さの割には
/   プログラムは小さくなったと（自分では）思いますが
/   そのせいで見にくくなってしまったので注意して下さい。
```

DTMへの招待

MIDIをめぐる環境

Kioi Makoto 紀尾井 誠

国産機種として見るとX68000のMIDI普及率はなかなかのものです。MIDIは内蔵音源や拡張音源ボードといった閉じた世界とは違い，音楽環境を開かれた世界へ誘います。現状と最新の情報を中心にX68000でのMIDI環境をまとめてみましょう。

市販ゲームのMIDI対応が進んできました。内蔵音源での音楽表現にも限界が見え始めました。こうしてますますMIDIに注目が集まりつつあります。ここではX68000をめぐるMIDI関係のお話をしていきます。これからMIDIを始めようという人は参考にしてください。

では，項目別に現状からまとめてみましょう。

MIDIボード

最近のシャープの広告にはMIDIボードが在庫僅少品として扱われているので，心配している方も多いのではないでしょうか。ご安心ください。どうやら，これまでのCZ-6BM1に代わってCZ-6BM1Aが発売される模様です。機能/性能は同じでお値段変わらず，完全コンパチ……では，なにが変わったのかというと，VCCI(電波障害関係の規格）に対応して電波ノイズが少なくなったのだそうです。

MIDIボードには純正品のほか，システムサコムからも発売されているものがあります。純正品との違いはテープシンク端子のないこととマニュアルにMIDIコントローラ関係の資料がまったくないことです。また，MIDIの標準プラグを差し込めるのはいいのですが，SCSIボードとの相性が悪く（コネクタが入りにくい)，近くバージョンアップが行われるのではという噂もあります。テープシンク端子は不要ですから，SCSIボード不要のSUPER/XVIユーザーにはお買い得なボードといえるでしょう。

ミキサー

なにはともあれ，X68000の音声出力はちゃんとしたオーディオに接続するのが基本です。内蔵スピーカで使っていると損をします。スピーカ付きのディスプレイもまあまあなのですが，MIDI出力をミキシングできないのでMIDIを導入したらオーディオ周りも考え直してみましょう。

内蔵音源とMIDIの出力をまとめるには，なんらかのミキサーが必要です。最近は4チャンネルのミキサーで1万円台ですから，楽器屋さんでちゃんとしたものを購入するのがもちろん最善といえます。

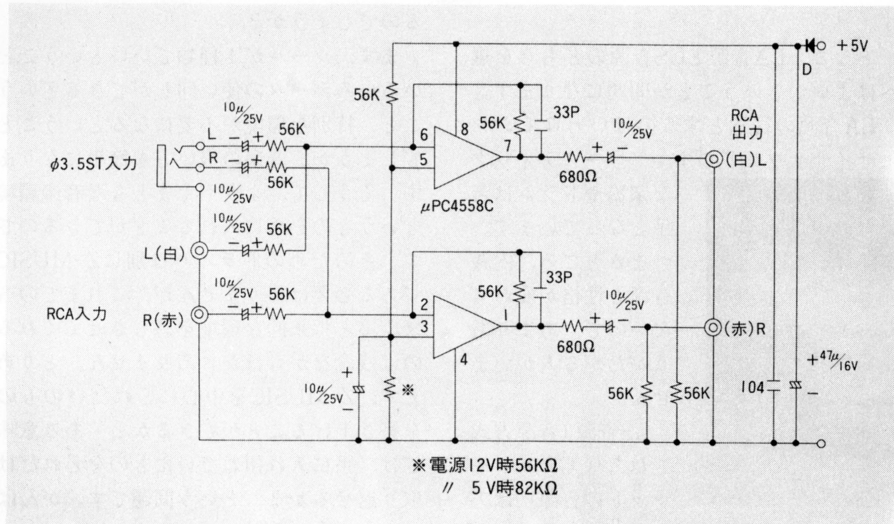
しかし，幸か不幸か，パソコンでMIDIを扱っている人の多くが音源モジュール1個しか扱っていないのも事実です。音源が1台ならば，もっと簡単にすませることもできます。

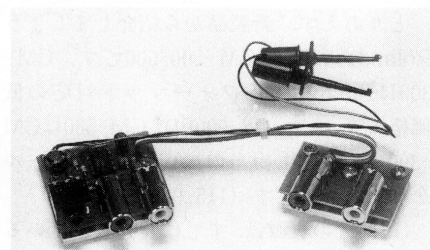
実際，結構多くの方が電波新聞社のミキシングケーブルを使用しているのではないでしょうか？　これはケーブルだけで安価かつ簡単に接続できますので，手軽にすませたい方にはいいでしょう。ただ，これは単に線をつないであるだけなので，楽器とX68000本体の双方に若干の負担をかけています。そういうのが気になる人にはおすすめできません。さらに内蔵音源の出力を本体全面のPHONES端子から取っているので，音質は若干落ちてしまいます（本体背面のAUDIO OUT端子のほうがノイズが少ない)。

MIDI楽器でもSC-55にはミキサーがついていますのでそのまま使えます。これならなにも考えることはありません。

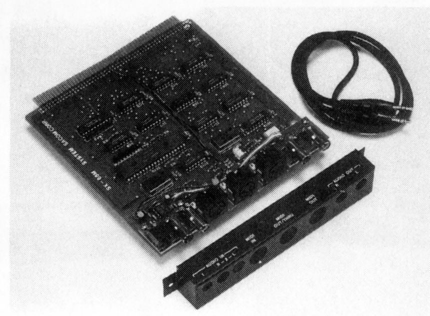
MIDIボードとしてサコム製のSX-68Mを使っている方ならサンミュージカルサー

図　X68000MIDIボード組み込みミニミキサー回路図





これをつけるだけ



ミキサー付きMIDIボード

ビスの改造キットが手軽でしょう。作業は小基板をネジ止めして，抵抗の足にクリップを引っ掛けて電源を取って，カバーを変えるだけです。ボードにはすでに穴が空いているので意外なほど簡単。

通販のみですが完成品も販売されるようです(SX-68MIXだそうだ)。こちらは12V電源を使ってさらに音質を上げたものとなっています。特にX68000SUPER/XVIユーザーで，いますぐMIDIを始めてみようと思っており，当面，音源はひとつしか使う予定がない，という人にはまさにうってつけのボードです。

ちゃんと背面のAUDIO OUTから出力を取るので音質もいいのですが，全面のPHONES端子と違ってコンピュータ側の音声出力レベルは変更できないので，MIDI楽器側の出力だけでバランスを取らなくてはならないのが，難点といえば難点です。Z-MISICシステムのサンプル曲「TWIN1.ZMS」(出たな!! ツインビー1面のテーマ)を演奏したときにはバランスを取るためにCM-64の音量ボリュームをいっぱいにしなければなりませんでした。

ちなみに，ハンダごての使える人なら，先ほどのキットを12V電源で使うように改造できます(電源線を基板に直付けして抵抗を1本変える)。詳しくは回路図を見てください(サンミュージカルサービス提供)。簡単な回路ですから自作するのもよいでしょう。

音源

とりあえず，新製品から紹介しましょう。Rolandの新音源CM-300/500です。CM-300はRolandのGSフォーマット対応の低価格モジュール(58,000円)。CM-500はCM-300にCM-32L相当のLA音源モジュールを加えたものです(115,000円)。

GS部分の音源は，すでに発売されているGS音源第1号のSC-55と同等と思われます。SC-55からリモコンや操作パネルなどを極力省いて低コスト化したものと考えてよいでしょう。

CM-500はいわゆる「MT-32」とGSの両方に対応したもので，CM-64シミュレーションモードを備えています。まだ，実際に試験をしていないのですが，一説にはSC-55のMT-32シミュレーションと違い十分実用になるレベルだと聞きます(LA音源部はCM-32Lとまったく同じ。エクスルーシブメッセージもOK)。これにCM-32Pを増設するとRS-PCMカードも使え，既存および将来のほとんどのデータに対応することができるはずです。

＊　＊　＊

最初の音源としてはCM-32L/64/500といった「MT-32と同等」な演奏ができるものが無難です。次いで，SC-55/CM-300といったGS音源になります。データの運用を考慮するとこれらのシリーズは無視できません。

もちろん，本格的に音楽制作するつもりの方なら自分の好みでなんでも選んでください。

本来はMIDIの転送速度で送られた信号は楽器側ですべて受け取られないといけないそうなのですが，多くの楽器はそれだけの性能を持っていません。人間が演奏する程度のデータなら楽器側のCPUはあまり高性能である必要がなかったことが原因でしょう。K1などでは音色切り換えをすると演奏が止まることがあります。M1などでもエクスクルーシブデータの送信時にはウェイトが必要です。

コンピュータの内蔵音源のように1ステップごとにパラメータを変えたりといった動作には到底ついてきません。RolandのMT-32,CM-32,SC-55などはそういった意味では非常によくできた音源だといえます。

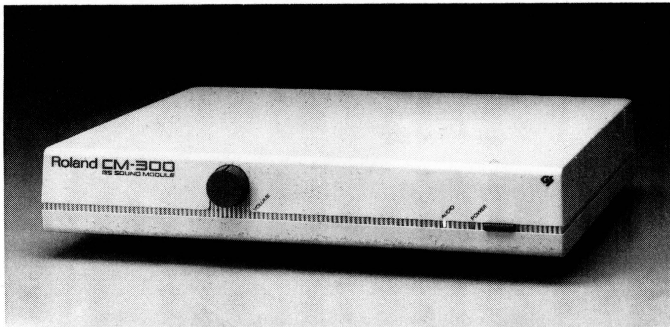
となるとLA音源とGS音源のどちらを選べばよいかということが問題になります。

LA音源の得意とするのはいわゆるシンセサイザサウンドです。アコースティックな楽器やドラムは苦手となっています。つまるところ，内蔵音源と性格が変わらないので，あまりありがたがる人がいません。

GS音源(音源方式ではなくGSフォーマットの音源)はアコースティックな音に強く，ドラムも強化されています。半面，シンセサイザではないので，音作りはかなり限定されています。あくまでも用意された音を選んで使うというのが基本スタイルです。今後音源方式が変わってもデータの互換性が保証される(ということになっている)のが最大の利点でしょう。

果たしてGSがLAに代わる標準となるかどうか，それが今後の動向の焦点となります。コムパックの「GROUP X」が早々とSC-55にも対応したのを始め，年末の大型ゲーム「出たな!! ツインビー」がMT-32とSC-55の両方に対応してきました。もちろん，Z-MUSICシステムでも機種別サポートの一部に加わっています。

ミュージックデータ資産は当然LAに軍配が上がりますし，市販ソフトの対応もほとんどLAのみです。しかしGSに対する注目度もかなり高いものがあり，予断を許しません。



CM-300

ドライバ

OPMDRV1.Xは事実上，闇に葬られ，MUSIC DRVに飽き足らず，オンラインソフトでは役不足，ということでZ-MUSICシステムが作成されました。少なくともOh!Xでは今後Z-MISICを中心に展開していくことになります。Oh!X LIVE in '92の主役はZ-MUSICシステムとなりますので，とりあえずなんらかの方法で入手してください。

さて，コンピュータでMIDI楽器を扱うのはどんなときでしょうか？ 純然たる音楽観賞，ゲームなどのBGM，作曲などの音楽制作，楽器内データの管理……。必ずしも，あらゆるデータを一元的に扱う必要はないのですが，扱えたならどんなメリットがあるのでしょうか？

まず，ツールが1種類でいいということ，いろいろデータの使い回しができるということ，特別な環境が不要になるということでしょうか。音楽の扱い方も簡単になります。そうして，ようやくまともな音楽環境というものを手に入れることができるのです。そのためのドライバは別にZ-MUSICである必要はありませんが，これまでの音楽環境と将来的な環境を感じさせてくれるのは残念ながらほかにありません。とりあえず，Z-MUSICを中心にどれだけのものを築き上げることができるか？ ある意味では，手に入れ損ねていたものをどれだけ取り返せるか？ という問題です。がんばってみましょう。

MIDIボードの使い方

MIDI出力方法論

Ushijima Takeo 牛島 健雄

MIDIを使ったプログラムやMIDIドライバを作りたい，という場合などに必要な知識をまとめてみました。あわせてMIDIボードを扱うためのライブラリを紹介します。RS-232CをMIDIとして扱う方法なども検討してみましょう。

「MIDIってなんですか？」いまどきこんなことを口に出そうものなら，たちまち白い目で見られてしまいます。MIDIは，いま，もっともトレンディなパソコンの使い方であり，それを知らないのはイカンですよ。

それほどまでに，DTM (Desk Top Music)と呼ばれるパソコンを中心としたMIDIによる音楽システムはメジャーになりました。我らがX68000でも例外ではなく，MIDIボードは飛ぶように売れ，その筋のネットには各種演奏データが氾濫しているようです。

かくいう私も，見事にはまったユーザーのひとりであり，よりよい音源を求めて楽器店をさまよい，暇を見つけてはMIDIキーボードを叩くまでになってしまいました。

さて，ここではDTMの中核を成すMIDI規格や音源についての話ではなく，影の立て役者でありながら，いままで日の目を浴びることのなかったパソコンと音源の接点であるMIDIボードを中心に『MIDIボード操縦法』と題して，その扱い方などに注目していきます。

MIDIボードって？

ところで，『MIDIボードはどんな機能を持っているのか』という問いに対して，明確な答えを提示できる人は案外少ないのではないでしょうか。

とはいうものの，パソコンで作曲・演奏をするのにMIDI規格やMIDIボードについての知識が必要であるのならば，これほどまでにDTMが流行ることもなかったでしょうから当然のことなのかもしれません。

もうひとつの理由として，MIDI規格については音源のマニュアルに明記されているものの (X68000に限らず)，MIDIボードに関する資料が不足しているということがあります。

純正のMIDIボードには，かろうじて参考になる程度の資料が付属していますが，これだけでMIDIボードを縦横無尽に使いこなせというのは絶望的でしょう。

「自作のソフトでMIDIを扱おうと思ってはみたものの，基本的な部分がよくわからないのであきらめてしまった」という例は，かなりあるものと思われます。

MIDIの扱い方の詳しい部分は，今回のソースリストを見ていただくなり，Oh!X BooksのZMUSIC.Xのソースを見て修得されるなりしていただくとして，まずMIDIボードの機能から紹介します。

MIDIボードの機能

MIDIボードは，MIDIデータ処理をハードウェアで行うためのものなのですが，それでは，扱う対象であるMIDIデータとはなんなのでしょう。

MIDIのデータ通信は，MIDI規格で定められた，31.25kbpsの非同期シリアル通信です。転送形式は，1バイトにつき，スタートビット/データビットがともに1ビット，データ長8ビットの計10ビットで行われます。

送受信回路は5mAの電流ループで，グランド短絡を避けるためにフォトカプラを使って，送信側と受信側を電気的に遮断してあります。実際のデータのやり取りは，MIDIケーブルと呼ばれるケーブルを通して，MIDI IN/OUT/THRUなどの5ピンDINコネクタ端子で行われます。

さて，これらMIDIケーブルを通してやり取りされるデータを，MIDIボードではどのように処理しているかというと，YAMAHAのYM3802 (MCS) というMIDIコミュニケーション用コントローラを使ってハード的に処理を行っています。

MIDIデータは，転送速度が31.25kbps，つまり1MHzの32分周の速度で送受信しなければならないのですから，単純に計算しても10MHzのX68000で320クロックの周期でデータをやり取りすることになります。ソフトウェアでは到底うまく対応することができないのは明らかです。そのためにハードウェア的にデータ処理を行って，CPUの負担を減らすのがYM3802の主な役割です。

また，YM3802は内部に送信用として16バイト，受信用として128バイトのバッファを独自に持っていますので，CPU側の処理はさらに軽減されることになります。

X68000でMIDIデータの送受信処理を行うためには，必要なときにYM3802の内部バッファに対して読み書き処理をするだけですみます。

YM3802のその他の機能として，14ビットの汎用タイマなど3つのタイマを内部に持っていますので，これらを有効に利用することで，MIDIドライバなどの処理も簡単に行うことができます。

入出力の実際

では，実際に入出力を行うにはどうすればよいのかという話に入ります。

MIDIボードのI/Oポートアドレスは，$EAFA01～$EAFA0Fに設定されており，38個のレジスタが配置されています。

これらはグループ別に分別されており，R01にレジスタの上位番号を，次にレジスタの下位番号にあたるアドレスにデータを書くことで，個々のレジスタにアクセスします。

これらのレジスタの役割や機能についてはここでは触れません。

MIDIをアクセスするためのサブルーチン・ライブラリをリスト1に示します。

基本的に，MIDIの入出力はR36 (受信データ) とR56 (送信データ) の2つで可能です。

入力例：

```
        move.b  #$03,$00EAFA03
        move.b  $00EAFA0D,d0
```

出力例：

```
move.b  #$05,$00EAFA03
move.b  d0,$00EAFA0D
```

もっとも簡単なものであれば，これだけで十分です。しかし，多くの場合，大量のデータを一度に入出力するわけですから，このままではYM3802のバッファは簡単にあふれてしまいます。あふれないようにバッファフルを監視していたとしても，バッファに空きができるまでは入出力を待たなければならないのですから，処理的にはかなりのウェイトがかかることになってしまい，速度低下の原因となります。

これを防ぐためには，ユーザープログラム内で入出力用の2次バッファを用意しておいて，MIDIボードのデータ受信割り込みと，送出バッファが空になったときに発生する割り込みを使って，割り込み内でデータのやり取りをするのが効果的です。

これならば，2次バッファの大きさ次第では，ソフトウェア上の負荷はかなり軽減されることが期待できます。

編注：ZMUSIC.XはMIDI処理に2次バッファを使用していない。2次バッファではCPUの負荷が軽いにもかかわらず音楽的テンポがずれることもあったため。

割り込みについて

MIDIボードで扱う割り込みには以下の7つの割り込みがあります。それぞれについて，簡単に解説します。

・IRQ0 MIDIリアルタイムメッセージ検出

受信データバッファの最古データが$F9～$FF，つまりリアルタイムメッセージになったときに割り込みを発生します。割り込み処理としては，これらのリアルタイムメッセージを各処理ブロックに送り，受信バッファをインクリメントする必要があります。

・IRQ1 MIDIクロック検出，またはCLICKカウンタ0カウント

この割り込みは，どちらかをソースとして選択しなければなりません。場合によって使い分けてください。

MIDIクロック検出は，受信バッファの最古データが$F8になったときに割り込みを発生します。または，CLICKカウンタが設定値をカウントし終わると割り込みを発生します。

・IRQ2 プレイバックカウンタ割り込み

プレイバックカウンタのカウント値が負になったときに割り込みが発生します。このカウンタは初期値の自動ロードを行いませんので，割り込み処理終了後は，プレイ

MIDIライブラリの説明

MIDIアクセス用ライブラリ(リスト1)内のコールについて，ここで若干の説明を加えておきます。それぞれのコールの引数，返り値などについては，ソースリストのコメントに詳しく書いてありますのでそちらを参照してください。

●MIDIボード関連

名前　Check_MIDI
機能　MIDIボードの存在をチェックするためのものです。プログラムの先頭で必ずチェックするようにしてください。

チェックには，MIDIボードが入っていないのに，MIDIボードのアドレスをアクセスするとバスエラーが出るのを利用しています。

名前　Init_MIDI
機能　MIDIボードを使えるように初期化するためのコールです。最初に必ず呼ぶようにしてください。初期化せずに入出力を行おうとした場合には，MIDIボードの動作は保証されません。

名前　SetUserVec
機能　MIDIボードが発生する割り込みをユーザーが利用する場合に，処理アドレスを登録するためのものです。ライブラリ内でソフトウェア的に，処理アドレスの呼び出しを行っています。ユーザー処理は"rts"で終了するようにしてください。これ以外の場合の動作は保証されません。

名前　SetVecMIDI
機能　MIDIボードの割り込みベクタ($80～$8E)をセットして，割り込みの発生を許可するためのものです。このコールが呼ばれた時点から，MIDIの入出力が可能になります。

名前　RmvVecMIDI
機能　MIDIボードの割り込みベクタを解除して，割り込みを禁止状態にします。このコールは，MIDIの使用終了を宣言するものと考えてください。

名前　MIDIput
機能　MIDIデータを1バイト出力するためのルーチンです。ボードの送信バッファにデータがなければ直接出力を行い，あればライブラリ内の送信用バッファ(256バイト)に溜めておいて，割り込みを使って自動的に送信します。

名前　MIDIget
機能　割り込みによって受信したデータを読み出します。ライブラリ内では，独自に受信用バッファ(256バイト)を確保して，割り込みで受信したものを溜めてあります。コールで読み出される値は，実際にはこのバッファに入っていたデータです。

名前　MIDIrmsg
機能　システムリアルタイムメッセージを送信バッファとは無関係に優先して出力します。システムリアルタイムメッセージとは，$F8～$FFまでの8種類のステータスバイトによる単独メッセージの総称です。演奏データの実時間性を高めるために，音程データなどのほかのデータなどに割り込んで送信することが許されています。

名前　MIDIGtimer
機能　YM3802の持つ14ビット汎用タイマの割り込み周期(8μs単位)を設定します。設定したカウント値をカウントし終わると，割り込みIRQ7を発生して，自動的に初期値をロードして再カウントを行います。

名前　MIDICtimer
機能　汎用タイマと同形の14ビットMIDIクロックカウンタの周期を設定します。このタイマは，設定値をカウントし終わると割り込み信号の代りに，MIDIクロックのタイミング信号を発生します。

名前　MIDICcounter
機能　内部MIDIクロックをカウントする7ビットプログラマブルカウンタであるCLICKカウンタの周期を設定します。設定値をカウントし終わると，CLICK端子に2msec幅のパルスを出力すると同時に，割り込み信号を発生し，初期値を自動ロードして，再カウントを行います。

名前　MIDIctrlRx
機能　受信バッファの動作を設定します。受信データ内のMIDIクロック($F8)をバッファに入れる前にチェックするMIDIクロックフィルタ機能とデータ内のデバイスIDとメーカーIDをチェックして受信するデータを制限できるアドレスハンタ機能の動作を設定できます。使い方によっては，ホストCPUの処理をかなり軽減することができます。

名前　MIDIIRQctrl
機能　7つあるMIDI割り込みのうちIRQ1とIRQ4の割り込み源を選択します。

名前　MIDIctrlS
機能　MIDIコントロールソースの選択を行います。アクティブセンシング($FE)の自動送出などのMIDIボード内の各部の動作をコントロールするための機能を設定します。

名前　MIDIreadClock
機能　レコーディングカウンタという8ビットの読み出し可能な固定カウンタをソフトウェアで桁上げをして16ビット自動カウントアップカウンタとしたリアルタイムクロックの読み出しをします。使い方によっては，非常に便利なものです。

名前　MIDIsetAHNT
機能　アドレスハンタの動作を設定します。アドレスハンタは，エクスクルーシブメッセージのように，機種によっては必要のないメッセージを受信処理しないようにするためのフィルタで，デバイスIDコードのみ，もしくはメーカーIDコードのチェックを組み合わせることで，受信処理がかなり楽になります。

●RS-232C関連

名前　Init_RS
機能　内蔵のRS-232Cポートを，MIDI規格で定められた，31.25kbps，スタートッビット/ストップビット1，ノンパリティ，データ長8ビットに設定するためのコールです。使用する前に必ず呼び出すようにしてください。

名前　SetVecRS
機能　RS-232C用の割り込みベクタを設定するためのものです。IOCS内の割り込み処理をそのまま使用するわけにはいきませんので，独自の割り込み処理ルーチンをセットします。これが，RS-232CでMIDI制御をするための開始宣言です。

名前　RmvVecRS
機能　RS-232C用の割り込みベクタを元に戻します。RS-232CによるMIDI制御の終了宣言です。

名前　RSput
機能　RS-232Cに1バイト出力するためのものです。MIDIputと同様の出力処理を行っています。送信用バッファは256バイトです。MIDIputとすり替えて使うことができます。

名前　RSget
機能　RS-232Cから1バイト入力するためのものです。MIDIgetと同様の入力処理を行っています。受信用バッファは256バイトです。MIDIgetとすり替えて使うことができます。

バックカウンタに加算ロードしてやることが必要です。

・IRQ3 レコーディングカウンタ割り込み

レコーディングカウンタの値が0になったときに割り込みを発生します。今回のMIDIライブラリでは,桁上げ処理にこの割り込みを使って，16ビットのカウンタに使用しています。

・IRQ4 ブレーク検出，またはオフライン検出

この割り込みも，どちらか割り込み源を選択しなければなりません。

ブレーク検出は，入力が2キャラクタ分（$2\times10/31250=0.64\mu$sec）の時間Lレベルのときに割り込み信号を発生します。また,オフライン検出は,300msecの間受信が行われなかった場合に割り込み信号を発生します。

・IRQ5 受信バッファ内データ有効

空の受信バッファにデータがセットされると割り込み信号を発生します。この割り込みを利用して，ユーザーが用意したバッファへデータを取り込むためには，割り込み処理終了時にはバッファを空にしなければなりません。

・IRQ6 送信バッファ空

送信用バッファが空になったときに割り込みを発生します。この割り込みを利用して送信を利用して割り込み発生時に，ユーザーが用意したバッファから1バイト送信することで，自動的にバッファ内のデータを送信することができます。

・IRQ7 汎用タイマ0カウント

汎用タイマに設定した値をカウントし終わると割り込みを発生し，自動的にカウント値のロードと再カウントを行います。

この割り込みを使えば，周期的に音符データを出力することも簡単にできます。

＊　＊　＊

挙げればきりがないですがざっとこんな感じです。

MIDIボードの拡張

X68000では，MIDIボードを2枚まで装備することができます。

2枚目のボードは，ジャンパースイッチを切り換えることで，アクセス用のI/Oポートアドレスをずらさなければなりません。

2枚目のボードのI/Oポートアドレスは，$EAFA10～$EAFA1Fに設定されており，ここを1枚目のときと同じようにアクセスすることでまったく同様に使えます。

しかし，注意しなければならないのは，MIDI割り込みベクタが，1枚目と2枚目とで重ならないように，ベクタオフセット値を変更しなければなりません。ベクタオフセット値は，R04で指定できます。X68000では，割り込みベクタ$6C以降が空きベクタとなっており,MIDIボードで使用できる割り込みベクタオフセットは，$80/$A0/$C0/$E0のいずれかに設定されています。

$80～$8Eを1枚目のMIDIボードがすでに使用していますので，2枚目の割り込みベクタは$A0～$AEなどに設定すればよいでしょう。あとは各人の好みの問題です。

1枚のボードでアクセスできるMIDIのチャンネル数は16チャンネルですから，2枚差すことで32チャンネルの演奏が可能になるわけです。

本体標準装備のMIDI?

最初の部分でMIDIは非同期シリアル通信であるということを述べました。PC-9801シリーズに浮気している人や，かなり勘のよい人はすでに気づいていると思いますが，本体に装備されているRS-232Cポートでも，MIDIデータ通信ができるのです。

RS-232Cポートを，MIDI規格で定められたとおりに設定しさえすれば，あとはMIDIボードと同様に扱えます。MIDIライブラリの後半部分がRS-232CでMIDIデータを扱うための処理です。

RS-232Cを制御しているデバイスはZ8530（SCC）というものなのですが，私は詳しい資料を見たことがありません。

さて，RS-232CでMIDIボードの代りができるといっても，ケーブルを直接つなぐわけにもいきません。出力だけでよいならば，抵抗とダイオードひとつで簡単に変換ケーブルを作ることができます。

自作するのが面倒であれば，PC-9801シリーズ用に発売されている，製品を購入すればそのまま利用できます。

とはいっても，対応しているソフトウェアがありませんので，残念ながらいまのところはなんの意味もないですね。

壮絶な演奏システムが可能に……

X68000 1台で，FM音源8ch＋AD PCM 1ch＋MIDI16ch＋RS16chで合計41chの演奏が可能です。MIDIボードをもう1枚追加すれば合計57chの演奏が可能です。

現在のところ41ch演奏までは検証済みで，演奏表現としてはもう十分なところまできていますが，あとはなにを演奏するかというデータの問題が残ります。

ここまでくれば，ナムコのシステム2などは目ではないし，オーケストラ顔負けの演奏もできるかもしれません。

編集室では，個人では絶対に実現不可能と思われる，さらに怪しい究極の演奏システム案が出ています。いつの機会にか紹介できるといいのですが……。

ここで示したMIDIアクセスライブラリはあくまでも一例ですので，各人の用途に合わせてアレンジするなり，参考にしていただくなど,活用していただければ幸いです。

参考文献

1) YM3802アプリケーションマニュアル,日本楽器
2) CZ-6BM1取扱説明書，シャープ

リスト1

```
* "midilib.s"
* #Source File No. 100.1.1991.10.24
*
* MIDIアクセス用ライブラリ
*       Copyright (c)1991 Ussy.
*
        .xdef   Check_MIDI      * MIDI ボード存在チェック
        .xdef   Init_MIDI       * MIDI ボード初期化
        .xdef   SetUserVec      * IRQ ユーザーコール設定
        .xdef   SetVecMIDI      * IRQ ベクタ設定
        .xdef   RmvVecMIDI      * IRQ ベクタ解除
        .xdef   MIDIput         * MIDI 1byte 出力
        .xdef   MIDIget         * MIDI 1byte 入力

        .xdef   MIDIrmsg        * Realtime Message 出力
        .xdef   MIDIGtimer      * 汎用タイマー・カウンタ設定
        .xdef   MIDICtimer      * クロック・カウンタ設定
        .xdef   MIDICcounter    * クリック・カウンタ設定
        .xdef   MIDIctrlRx      * 受信バッファ制御
        .xdef   MIDIIRQctrl     * IRQ1/IRQ4 割り込みモード設定
        .xdef   MIDIctrlS       * リアルタイム・メッセージ制御
        .xdef   MIDIreadClcok   * リアルタイム・クロック読み出し
        .xdef   MIDIsetAHNT     * アドレスハンタの設定

        .xdef   Init_RS         * RS ポート初期化
        .xdef   SetVecRS        * RS 割り込みベクタ設定
        .xdef   RmvVecRS        * RS 割り込みベクタ解除
        .xdef   RSput           * RS 1byte 出力
        .xdef   RSget           * RS 1byte 入力

        .include        doscall.mac
        .include        iocscall.mac
*
* マクロ
*
MOUT                    .macro  data,reg
  .if ((reg/16).eq.255)
        st      $00eafa03
  .elseif ((reg/16).eq.0)
        sf      $00eafa03
  .else
        move.b  #(reg/16),$00eafa03
  .endif
  .if (data.eq.0)
        sf      $00eafa01+(reg.and.%1111)*2
  .elseif (data.eq.255)
        st      $00eafa01+(reg.and.%1111)*2
  .else
        move.b  #data,$00eafa01+(reg.and.%1111)*2
  .endif
        .endm
*
* MIDIボードアドレス
*
```

```
RGR                 equ     $00EAFA03
GRP2                equ     $00EAFA05
GRP3                equ     $00EAFA07
GRP4                equ     GRP3+2
GRP5                equ     GRP4+2
GRP6                equ     GRP5+2
GRP7                equ     GRP6+2
NULL                equ     0
WREG                set     a6
          .text
*
* MIDIボードチェック
*
* 引数      なし
* 返り値  d0.l     midi board exist         : d0.l = 0
*                   midi board not exist     : d0.l = -1
*
reglist             set     d1-d7/a0-a6
Check_MIDI:
          move.w    SR,-(sp)
          movem.l   reglist,-(sp)
          ori.w     #$0700,SR
          movea.l   #8,a1
          lea       abort_ssp(pc),a0          * アボートベクタ設定
          move.l    SP,(a0)+
          move.l    (a1),(a0)
          lea       x_chk_midi(pc),a0
          move.l    a0,(a1)
          tst.b     $00eafa01                 * バスエラー?
          moveq     #0,d0
          bra       r_chk_midi
x_chk_midi:
          move.l    abort_ssp(pc),SP
          moveq     #-1,d0
r_chk_midi:
          move.l    abort_vec(pc),(a1)
          movem.l   (sp)+,reglist
          rte
*
* MIDIイニシャライズ
*
* 引数      なし
* 返り値  なし
*
reglist             set     d0-d7/a0-a6
Init_MIDI:
          move.w    SR,-(sp)
          movem.l   reglist,-(sp)
          ori.w     #$0700,SR
          lea       RGR,a0
          move.b    #%1000_0000,(a0)          * イニシャルリセット
wait:
          moveq     #10,d7
p_wait:
          nop                                 * 8 ms 以上のウェイト
          dbra      d7,p_wait
p_init_mdidi:
          sf        (a0)                      * イニシャルリセット解除
          MOUT      %1000_0000,$04            * ベクタオフセット設定
          sf        RGR
          sf        GRP6                      * 割り込み禁止
          MOUT      %0000_0010,$66            * CLKM = 1MHz
          move.w    #$83e8,d0                 * 1000*8 μs
          bsr       MIDIGtimer                * 汎用タイマー
          move.w    #$87d0,d0                 * 2000*8 μs
          bsr       MIDICtimer                * クロックタイマー
          move.b    #$18,d0                   * 24 MIDI Clock
          bsr       MIDICcounter              * クリックカウンタ
          MOUT      %1001_0100,$65
          MOUT      %1000_0000,$55            * 送信バッファクリア,送信禁止
Tx_wait:
          btst.b    #0,GRP4
          bne       Tx_wait
          MOUT      %0000_1000,$44            * 送信ボーレート設定
          MOUT      %0000_0000,$45            * 送信パラメータ設定
          MOUT      %1001_0000,$35            * 受信バッファクリア,受信禁止
Rx_wait:
          btst.b    #0,GRP4
          bne       Rx_wait
          MOUT      %0000_1000,$24            * 受信ボーレート設定
          MOUT      %0000_0000,$25            * 受信パラメータ設定
          moveq     #0,d0
          bsr       MIDIIRQctrl               * 割り込みモード制御
          move.b    #%0001_0010,GRP3          * R03
          move.b    #%0010_0011,d0
          bsr       MIDIctrlS                 * リアルタイムメッセージ制御
          MOUT      %0000_0000,$94            * 外部I/Oポートの入出力の設定
          MOUT      %0000_0001,$35            * 受信許可
          MOUT      %0000_0001,$55            * 送信許可
          lea       MidiWORK(pc),WREG
          clr.w     readTx(WREG)
          clr.w     writeTx(WREG)
          clr.w     readRx(WREG)
          clr.w     writeRx(WREG)
          sf        overflow(WREG)
          sf        bufffull(WREG)
          lea       VecTable(pc),a0           * ユーザーベクタの初期化
          lea       MIDInoexec(pc),a1
          moveq     #5-1,d7
s_init_midi:
          move.l    a1,(a0)+
          dbra      d7,s_init_midi
r_init_midi:
          movem.l   (sp)+,reglist
          rte
*
* ユーザー割り込みベクタの設定
*
* 引数      a0.l     ベクタテーブル・アドレス
*            0(a0)   ; IRQ0 処理ルーチン
*            4(a0)   ; IRQ1 処理ルーチン
*            8(a0)   ; IRQ2 処理ルーチン
*           12(a0)   ; IRQ4 処理ルーチン
*           16(a0)   ; IRQ7 処理ルーチン
* 注)使わない処理のアドレスは 0 にしておくこと。
*
* 返り値  なし
*
reglist             set     d0-d7/a1-a6
SetUserVec:
          move.w    SR,-(sp)
          movem.l   reglist,-(sp)
          ori.w     #$0700,SR
          sf        RGR                       * R06
          sf        GRP6                      * 割り込み禁止
          lea       VecTable(pc),a1           * ユーザーベクタTable
          lea       MIDInoexec(pc),a2
          moveq     #5-1,d7
SUV000:
          move.b    (a0)+,-(sp)
          move.w    (sp)+,d0
          move.b    (a0)+,d0
          swap      d0
          move.b    (a0)+,-(sp)
          move.w    (sp)+,d0                  * 要するに・・・
          move.b    (a0)+,d0                  * move.l (a1)+,d0
          tst.l     d0                        * NULL ならば 空処理
          bne       SUV010
          move.l    a2,d0
SUV010:
          move.l    d0,(a1)+                  * コールを登録
          dbra      d7,SUV000
          sf        RGR                       * R06
          st        GRP6                      * 割り込み許可
r_SetUserVec:
          movem.l   (sp)+,reglist
          rte
*
* 割り込みベクタ設定
*
* 引数      なし
* 返り値  なし
*
reglist             set     d7/a0-a4
SetVecMIDI:
          move.w    SR,-(sp)
          movem.l   reglist,-(sp)
          ori.w     #$0700,SR
          sf        RGR                       * R06
          sf        GRP6                      * 割り込み禁止
          lea       MIDIvectbl(pc),a0
          lea       MIDIvec(pc),a1
          lea       $200.w,a2                 * INT $80
          movea.l   a0,a3
          moveq     #8-1,d7
p_SetVecMIDI:
          move.w    (a0)+,d0
          lea       (a3,d0.w),a4              * ベクタセット
          move.l    (a2),(a1)+
          move.l    a4,(a2)
          addq.l    #8,a2
          dbra      d7,p_SetVecMIDI
r_SetVecMIDI:
          sf        RGR                       * R06
          st        GRP6                      * 割り込み許可
          moveq     #0,d0
          movem.l   (sp)+,reglist
          rte
*
* 割り込みベクタ解除
*
* 引数      なし
* 返り値  なし
*
reglist             set     d7/a0-a1
RmvVecMIDI:
          move.w    SR,-(sp)
          movem.l   reglist,-(sp)
          ori.w     #$0700,SR
          lea       MIDIvec(pc),a0
          tst.l     (a0)
          beq       e_RmvVecMIDI
          sf        RGR                       * R06
          sf        GRP6                      * 割り込み禁止
          lea       $200.w,a1
          moveq     #8-1,d7
p_RmvVecMIDI:
          move.l    (a0)+,(a1)                * ベクタ初期化
          addq.l    #8,a1
          dbra      d7,p_RmvVecMIDI
          lea       MIDIvec(pc),a0
          moveq     #7,d0
s_RmvVecMIDI:
          clr.l     (a0)+                     * ベクタバッファ・クリア
          dbra      d0,s_RmvVecMIDI
          moveq     #0,d0
r_RmvVecMIDI:
          movem.l   (sp)+,reglist
          rte
e_RmvVecMIDI:
          moveq     #-1,d0                    * エラー発生
          bra       r_RmvVecMIDI
*
* MIDI送信
*
* 引数      d0.b     送信データ
* 返り値  d0.l     ステータス
*                     0 ; 正常送信
*                    -1 ; バッファフル
*
reglist             set     d1/a0/a6
MIDIput:
          move.w    SR,-(sp)
          ori.w     #$0700,SR
          movem.l   reglist,-(sp)
          lea       MidiWORK(pc),WREG
          tst.b     bufffull(WREG)            * バッファフルか?
          bne       Txbufffull
          move.b    #5,RGR                    * R54
          btst      #7,GRP4                   * 送信バッファが空か?
          beq       MP000
          btst      #6,GRP2                   * 割り込みが発生している?
          beq       p_MIDIput
MP000:
          lea       Txbuff(WREG),a0           * バッファへストア
          move.w    writeTx(WREG),d1
          move.b    d0,0(a0,d1.w)
          addq.w    #1,d1
          andi.w    #256-1,d1
          move.w    d1,writeTx(WREG)
          cmp.w     readTx(WREG),d1
          seq       bufffull(WREG)            * バッファフル?
r_MIDIput:
          moveq     #0,d0
          movem.l   (sp)+,reglist
          rte
p_MIDIput:
          move.b    d0,GRP6                   * 直接出力
          bra       r_MIDIput
Txbufffull:
          moveq     #-1,d0                    * バッファフル
          movem.l   (sp)+,reglist
          rte
*
* MIDIリアルタイム・メッセージ送信
*
* 引数      d0.b     送信データ
*   <bit> 76543210          ; P は1で送出、0で送出しない
*         P-------          ; リアルタイム・メッセージ用 FIFO-ITx
*         -P------          ; SYNC コントローラ
*         --P-----          ; クリック・カウンタ
*         ---P----          ; プレイバック・カウンタ
*         ----P---          ; レコーディング・カウンタ
*         -----000          ; $F8(ミディ・クロック) 送出
*         -----001          ; $F9                   送出
*         -----010          ; $FA(スタート)         送出
*         -----011          ; $FB(ストップ)         送出
*         -----100          ; $FC(コンティニュー)   送出
*         -----101          ; $FD                   送出
*         -----110          ; (禁止)
*         -----111          ; (禁止)
```

▶満開の電子ちゃんの母は娘に負けないくらいボケだった。父や弟の今後の活躍に期待します。
関口　智博(16)富山県

```
* 返り値  なし
*
MIDIrmsg:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            movem.l d0/a6,-(sp)
            move.b  #1,RGR
            move.b  d0,GRP5                     * R15
            andi.b  #%0000_1111,d0
            cmpi.b  #%0000_1010,d0              * $FA か?
            bne     r_MIDIrmsg
            lea     VecWORK(pc),WREG
            clr.w   0(WREG)                     * Realtime Clock初期化
r_MIDIrmsg:
            movem.l (sp)+,d0/a6
            rte
*
* MIDI受信
*
* 引数    なし
* 返り値  d0.b = 0~127      ; 受信データ
*              = -1         ; 最新データなし
*              = -2         ; オーバーフロー発生
*
reglist             set     d1-d7/a0-a6
MIDIget:
            move.w  SR,-(sp)
            movem.l reglist,-(sp)
            lea     MidiWORK(pc),WREG
            tst.b   overflow(WREG)              * オーバーフロー発生
            bne     overflwMIDI
            move.w  writeRx(WREG),d0
            cmp.w   readRx(WREG),d0
            beq     nodataMIDI
            lea     Rxbuff(WREG),a0
            move.w  readRx(WREG),d1             * バッファから読み出し
            moveq   #0,d0
            move.b  (a0,d1.w),d0
            addq.w  #1,d1
            andi.w  #256-1,d1
            move.w  d1,readRx(WREG)
r_MIDIget:
            movem.l (sp)+,reglist
            rte
nodataMIDI:
            moveq   #-1,d0                      * 最新データなし
            bra     r_MIDIget
overflwMIDI:
            moveq   #-2,d0                      * オーバーフロー発生
            clr.w   readRx(WREG)                * 読みだしポインタ初期化
            clr.w   writeRx(WREG)               * 書き込みポインタ初期化
            sf      overflow(WREG)              * オーバーフローフラグ初期化
            MOUT    %1100_0001,$35              * MIDIボードバッファ初期化
            bra     r_MIDIget
*
* MIDI汎用タイマー設定
*
* 引数    d0.w      タイマー値(8μsec単位)
*   <bit15> = 1     ; ロード直後からのカウント要求
*           = 0     ; ロードのみ行なう
* 返り値  なし
*
MIDIGtimer:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.l  d0,-(sp)
            move.b  #8,RGR
            move.b  d0,GRP4                     * R84
            lsr.w   #8,d0
            move.b  d0,GRP5                     * R85
r_MIDIGtimer:
            move.l  (sp)+,d0
            rte
*
* MIDIクロックタイマー設定
*
* 引数    d0.w      タイマー値(8μsec単位)
*   <bit15> = 1     ; ロード直後からのカウント要求
*           = 0     ; ロードのみ行なう
* 返り値  なし
*
MIDICtimer:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.l  d0,-(sp)
            move.b  #8,RGR
            move.b  d0,GRP6                     * R86
            lsr.w   #8,d0
            move.b  d0,GRP7                     * R87
r_MIDICtimer:
            move.l  (sp)+,d0
            rte
*
* MIDIクリックカウンター設定
*
* 引数    d0.b      カウンター値(MIDIクロック単位)
*   <bit7> = 1      ; ロード直後からのカウント要求
*          = 0      ; ロードのみ行なう
* 返り値  なし
*
MIDICcounter:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.l  d0,-(sp)
            move.b  #6,RGR
            bset    #7,d0
            move.b  d0,GRP7                     * R67
r_MIDICcounter:
            move.l  (sp)+,d0
            rte
*
* 受信バッファ制御
*
* 引数    d0.b
*   <bit> 76543210
*         ******0-          ; MIDIクロック・フィルター動作を許可
*         ******1-          ;        〃           を禁止
*         ******-0          ; アドレスハンタ動作を許可
*         ******-1          ;        〃      を禁止
*
MIDIctrlRx:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.l  d7,-(sp)
            moveq   #0,d7
            btst    #1,d0                       * MIDI clock ﾌｨﾙﾀｰ
            beq     MIDIctrlRx0
            ori.b   #%0000_0010,d7
MIDIctrlRx0:
            btst    #0,d0                       * address hunter
            beq     MIDIctrlRx1
            ori.b   #%0001_0000,d7
MIDIctrlRx1:
            move.b  #3,RGR                      * R35
            move.b  d7,GRP5                     * 受信バッファ動作設定
r_MIDIctrlRx:
            move.l  (sp)+,d7
            rte
*
* 割り込みモード・コントロール
*
* 引数    d0.b
*   <bit> 76543210
*         ******0-          ; IRQ4の割り込み源はオフライン検出
*         ******1-          ;     〃      ブレーク検出
*         ******-0          ; IRQ1の割り込み源は Click Counter ゼロ
*         ******-1          ;     〃        MIDIクロック検出
* 返り値  なし
*
MIDIIRQctrl:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.l  d7,-(sp)
            move.b  #%0000_0010,d7
            btst    #1,d0                       * IRQ4 割り込み源
            beq     MIDIIRQc0
            ori.b   #%0000_0100,d7
MIDIIRQc0:
            btst    #0,d0                       * IRQ1 割り込み源
            beq     MIDIIRQc1
            ori.b   #%0000_1000,d7
MIDIIRQc1:
            sf      RGR                         * R05
            move.b  d7,GRP5                     * 割り込みモード設定
r_MIDIIRQctrl:
            move.l  (sp)+,d7
            rte
*
* MIDIコントロール・ソースの選択
*
* 引数    d0.b
*   <bit> 76543210
*         0-------          ; Other Control Message 手動送出
*         1-------          ; Other Control Message 自動送出(通常)
*         --0-----          ; Active Sence 自動送信禁止
*         --1-----          ; Active Sence 自動送出許可
*         ---0----          ; Midi-Clock 自動送出禁止
*         ---1----          ; Midi-Clock 自動送出許可(通常)
*         ----0---          ; System Realtime 自動送出禁止
*         ----1---          ; System Realtime 自動送出許可(通常)
*         -----0--          ; 内部 Midi クロック源を選択(通常)
*         -----1--          ; ホストCPU の R15 書き込みタイミング
*         ------01          ; 受信データ中の Clock Message
*         ------10          ; テープSYNC 入力
*         ------11          ; Midi Clock Timer(通常)
* 返り値  なし
*
MIDIctrlS:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            movem.l d0/a6,-(sp)
            lea     VecWORK(pc),WREG
            move.b  d0,3(WREG)
            andi.b  #%0011_1111,d0
            ori.b   #%0001_1000,d0
            move.b  #1,RGR                      * R14
            move.b  d0,GRP4                     * ソースの選択
r_MIDIctrlS:
            movem.l (sp)+,d0/a6
            rte
*
* リアルタイム・クロック読み出し
*
* 引数    なし
* 返り値  d0.w      内部クロック読み出し値
*
MIDIreadClcok:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            movem.l d1/a6,-(sp)
            lea     VecWORK(pc),WREG
            moveq   #0,d0
            move.w  0(WREG),d0                  * upper word
            move.b  #7,RGR                      * R74
            move.b  GRP4,d0                     * Recording Counter
            bmi     r_MIDIrClock
            move.b  GRP2,d1                     * R02
            andi.b  #%0000_1000,d1              * オーバーフローした?
            beq     r_MIDIrClock
            addi.w  #$0100,d0                   * Counter increment
r_MIDIrClock:
            movem.l (sp)+,a6/d1
            rte
*
* アドレスハンタの設定
*
* 引数    d0.b
*   <bit> 76543210
*         0-------          ; ﾒｰｶｰID だけをチェックする
*         1-------          ; ﾒｰｶｰID と ﾃﾞﾊﾞｲｽID の両方をチェックする
*         -*******          ; メーカーIDコード
*         d1.b
*   <bit> 76543210
*         0-------          ; 指定した ﾃﾞﾊﾞｲｽID のみ比較する
*         1-------          ; $7F でもﾃﾞﾊﾞｲｽIDｺｰﾄﾞ一致として扱う
*         -*******          ; デバイスIDコード
* 返り値  なし
*
MIDIsetAHNT:
            move.w  SR,-(sp)
            ori.w   #$0700,SR
            move.b  #2,RGR
            move.b  d0,GRP6             * R26
            move.b  d1,GRP7             * R27
r_MIDIsetAHNT:
            rte
*
* ヌルイベント(割り込み処理)
*
* 引数    なし
* 返り値  なし
*
MIDInoexec:
            rts
*
* MIDIリアルタイム・メッセージ検出
*
* 引数    なし
* 返り値  なし
*
reglist             set     d0-d7/a0-a6
MIDIIRQ0:
            movem.l reglist,-(sp)
            move.b  #%0000_0001,GRP3            * 割り込みクリア
            move.b  #1,RGR                      * R16
            move.b  GRP6,d0
            lea     VecWORK(pc),WREG
            tst.b   3(WREG)                     * 自動送出か?
            bpl     p_MIDIIRQ0
            ori.b   #%1111_1000,d0
            move.b  d0,GRP5
            move.b  #1,GRP7                     * FIFO-Rxをインクリメント
```

▶友人の車で事故を起こして借金を抱えてしまいました。今月の食費は5千円。最近私の主食はパンの耳です。
中村　学(20)石川県

```
        lea     VecTable(pc),WREG
        pea     r_MIDIIRQ0(pc)              * リターンアドレス
        move.l  0(WREG),-(sp)
        rts
p_MIDIIRQ0:
        lea     VecTable(pc),WREG
        pea     s_MIDIIRQ0(pc)              * リターンアドレス
        move.l  0(WREG),-(sp)
        rts
s_MIDIIRQ0:
        move.b  #1,RGR                      * R17
        move.b  #1,GRP7                     * FIFO-Rxをインクリメント
r_MIDIIRQ0:
        movem.l (sp)+,reglist
        rte
*
* MIDIクロック検出
*
* 引数    なし
* 返り値  なし
*
MIDIIRQ1:
        movem.l reglist,-(sp)
        move.b  #%0000_0010,GRP3            * 割り込みクリア
        lea     VecTable(pc),WREG
        pea     r_MIDIIRQ1(pc)              * リターンアドレス
        move.l  4(WREG),-(sp)
        rts
r_MIDIIRQ1:
        movem.l (sp)+,reglist
        rte
*
* プレイバック・カウンタ割り込み
*
* 引数    なし
* 返り値  なし
*
MIDIIRQ2:
        movem.l reglist,-(sp)
        moveq   #0,d0
        lea     VecTable(pc),WREG
        move.l  WREG,-(sp)
        pea     s_MIDIIRQ2(pc)              * リターンアドレス
        move.l  8(WREG),-(sp)
        rts
s_MIDIIRQ2:
        movea.l (sp)+,WREG
        move.b  #7,RGR                      * R76
        move.b  d0,GRP6                     * 次のインターバル値
        lsr.w   #8,d0
        move.b  d0,GRP7                     * R77
        move.b  #%0010_0000+2,GRP5          * 補間レイト := 2
        move.b  #%0000_0100,GRP3            * 割り込みクリア
r_MIDIIRQ2:
        movem.l (sp)+,reglist
        rte
*
* レコーディング・カウンタ割り込み
*
* 引数    なし
* 返り値  なし
*
MIDIIRQ3:
        movem.l reglist,-(sp)
        move.b  #%0000_1000,GRP3            * 割り込みクリア
        lea     VecWORK(pc),WREG
        addq.b  #1,0(WREG)                  * Counter increment
r_MIDIIRQ3:
        movem.l (sp)+,reglist
        rte
*
* オフライン割り込み
*
* 引数    なし
* 返り値  なし
*
MIDIIRQ4:
        movem.l reglist,-(sp)
        move.b  #%0001_0000,GRP3            * 割り込みクリア
        lea     VecTable(pc),WREG
        pea     r_MIDIIRQ4(pc)              * リターンアドレス
        move.l  12(WREG),-(sp)
        rts
r_MIDIIRQ4:
        movem.l (sp)+,reglist
        rte
*
* MIDI受信割り込み
*
* 引数    なし
* 返り値  なし
*
reglist         set     d0-d1/a0/a6
MIDIIRQ5:
        ori.w   #$0700,SR
        movem.l reglist,-(sp)
        lea     MidiWORK(pc),WREG
        move.b  #3,RGR
RxD000:
        move.b  GRP4,d0                     * 受信ステータスチェック
        bpl     r_Rxirpt                    * R34
        move.b  GRP6,d0                     * データ取得(R36)
        cmpi.b  #$FE,d0                     * アクティブセンシングか?
        beq     RxD000
        lea     Rxbuff(WREG),a0
        move.w  writeRx(WREG),d1
        move.b  d0,(a0,d1.w)
        addq.w  #1,d1
        andi.w  #256-1,d1
        move.w  d1,writeRx(WREG)
        cmp.w   readRx(WREG),d1
        beq     bufoverMIDI
        bra     RxD000
r_Rxirpt:
        move.b  #%0010_0000,GRP3            * 割り込みクリア
        movem.l (sp)+,reglist
        rte
bufoverMIDI:
        move.b  GRP4,d0
        bpl     r_bufoverMIDI
        move.b  GRP6,d0                     * データ切り捨て
        bra     bufoverMIDI
r_bufoverMIDI:
        st      overflow(WREG)              * オーバーフロー発生
        bra     r_Rxirpt
*
* 送信バッファ空
*
* 引数    なし
* 返り値  なし
*
reglist         set     d0-d1/a0/a6
MIDIIRQ6:
        ori.w   #$0700,SR
        movem.l reglist,-(sp)
        move.b  #%0100_0000,GRP3            * 割り込みクリア
        lea     MidiWORK(pc),WREG
        sf      bufffull(WREG)
        move.w  readTx(WREG),d0
        cmp.w   writeTx(WREG),d0
        beq     r_TxData
        move.b  #5,RGR                      * グループ5を選択
TxD000:
        btst    #6,GRP4                     * 送信バッファフル?
        beq     r_TxData                    * R54
        lea     Txbuff(WREG),a0
        move.b  0(a0,d0.w),GRP6             * 出力
        addq.w  #1,d0
        andi.w  #256-1,d0
        move.w  d0,readTx(WREG)
r_TxData:
        movem.l (sp)+,reglist
        rte
*
* 汎用タイマー 0カウント割り込み
*
* 引数    なし
* 返り値  なし
*
reglist         set     d0-d7/a0-a6
MIDIIRQ7:
        move.w  SR,-(sp)
        ori.w   #$0700,SR
        movem.l reglist,-(sp)
        move.b  #5,RGR                      * 送出禁止
        sf      GRP5                        * R55
        move.b  #%1000_0000,GRP3            * 割り込みクリア
        lea     VecTable(pc),WREG
        pea     r_MIDIIRQ7(pc)              * リターンアドレス
        move.l  16(WREG),-(sp)
        rts
r_MIDIIRQ7:
        move.b  #$05,RGR                    * 送出許可(R55)
        move.b  #%0000_0001,GRP5
        movem.l (sp)+,reglist
        move.w  (sp)+,SR
        rte
*
* RSベクタセット
*
* 引数    なし
* 返り値  なし
*
reglist         set     d1-d7/a0-a6
SetVecRS:
        move.w  SR,-(sp)
        movem.l reglist,-(sp)
        ori.w   #$0700,SR
        lea     $160.w,a0
        lea     RSvec(pc),a1
        moveq   #7,d0
p_SetVecRS:
        move.l  (a0)+,(a1)+
        dbra    d0,p_SetVecRS
        lea     $160.w,a0
        lea     TxDataRS(pc),a1
        move.l  a1,(a0)+
        move.l  a1,(a0)+
        lea     ExtStat(pc),a1
        move.l  a1,(a0)+
        move.l  a1,(a0)+
        lea     RxDataRS(pc),a1
        move.l  a1,(a0)+
        move.l  a1,(a0)+
        lea     SpecialRx(pc),a1
        move.l  a1,(a0)+
        move.l  a1,(a0)+
        bsr     Init_RS
        moveq   #0,d0
r_SetVecRS:
        movem.l (sp)+,reglist
        rte
*
* RSベクタリリース
*
* 引数    なし
* 返り値  ステータス(負数はエラー)
*
reglist         set     d1-d7/a0-a6
RmvVecRS:
        move.w  SR,-(sp)
        movem.l reglist,-(sp)
        ori.w   #$0700,SR
        lea     RSvec(pc),a0
        tst.l   (a0)
        beq     r_RmvVecRS
        move.b  #9,$00e98005
        move.b  #$89,$00e98005
        lea     $160.w,a1
        moveq   #7,d0
p_RmvVecRS:
        move.l  (a0)+,(a1)+
        dbra    d0,p_RmvVecRS
        lea     RSvec(pc),a0
        moveq   #7,d0
s_RmvVecRS:
        clr.l   (a0)+
        dbra    d0,s_RmvVecRS
        moveq   #0,d0
r_RmvVecRS:
        movem.l (sp)+,reglist
        rte
e_RmvVecRS:
        moveq   #-1,d0
        bra     r_RmvVecRS
*
* RSポート初期化
*
* 引数    なし
* 返り値  なし
*
reglist         set     d0-d7/a0-a6
Init_RS:
        move.w  SR,-(sp)
        movem.l reglist,-(sp)
        ori.w   #$0700,SR
        move.b  $00e98005,d0                * ダミー入力
        lea     RSdata(pc),a0
        moveq   #$1f,d0
p_Init_RS:
        move.b  (a0)+,$00e98005             * 初期化データ出力
        dbra    d0,p_Init_RS
        lea     RsWORK(pc),WREG
        clr.w   readTx(WREG)
        clr.w   writeTx(WREG)
        clr.w   readRx(WREG)
        clr.w   writeRx(WREG)
        sf      overflow(WREG)
        sf      bufffull(WREG)
r_Init_RS:
        movem.l (sp)+,reglist
        rte
```

▶最近メモリとかが安くなっているようだ（X68000用）。しかし自分が買ったあとにすぐ，2万も3万も安くなるとはとっても悔しい。まあ，安くなることはいいことだけど。
相馬　信隆(26)神奈川県

```
*
* RS送信
*
* 引数    d0.b     送信データ
* 返り値  d0.l     ステータス
*                   0 ; 正常送信
*                  -1 ; バッファフル
*
reglist             set      d1/a0/a6
RSput:
            move.w  SR,-(sp)
            movem.l reglist,-(sp)
            ori.w   #$0700,SR
            lea     RsWORK(pc),WREG
            tst.b   bufffull(WREG)              * バッファフルか?
            bne     e_RSput
            move.w  writeTx(WREG),d1            * 最新データあり?
            cmp.w   readTx(WREG),d1
            bne     RP000
            btst    #2,$00e98005                * バッファが空?
            bne     p_RSput
RP000:
            lea     Txbuff(WREG),a0
            move.b  d0,0(a0,d1.w)               * バッファにストア
            addq.w  #1,d1
            andi.w  #$ff,d1
            move.w  d1,writeTx(WREG)
            cmp.w   readTx(WREG),d1
            seq     bufffull(WREG)
r_RSput:
            moveq   #0,d0
            movem.l (sp)+,reglist
            rte
p_RSput:
            move.b  d0,$00e98007                * 直接出力
            bra     r_RSput
e_RSput:
            moveq   #-1,d0                      * バッファフル
            movem.l (sp)+,reglist
            rte
*
* RS受信
*
* 引数    なし
* 返り値  d0.b = 0~127     ; 受信データ
*              = -1        ; 最新データなし
*              = -2        ; オーバーフロー発生
*
reglist             set      d1-d7/a0-a6
RSget:
            move.w  SR,-(sp)´
            movem.l reglist,-(sp)
            lea     RsWORK(pc),WREG
            tst.b   overflow(WREG)              * オーバーフロー発生
            bne     overflwRS
            move.w  writeRx(WREG),d0            * 最新データあり?
            cmp.w   readRx(WREG),d0
            beq     nodataRS
            lea     Rxbuff(WREG),a0
            move.w  readRx(WREG),d1
            moveq   #0,d0
            move.b  (a0,d1.w),d0                * バッファから取得
            addq.w  #1,d1
            andi.w  #256-1,d1
            move.w  d1,readRx(WREG)
r_RSget:
            movem.l (sp)+,reglist
            rte
nodataRS:
            moveq   #-1,d0                      * 最新データなし
            bra     r_RSget
overflwRS:
            moveq   #-2,d0                      * オーバーフロー発生
            clr.w   readRx(WREG)                * 読みだしポインタ初期化
            clr.w   writeRx(WREG)               * 書き込みポインタ初期化
            sf      overflow(WREG)              * オーバーフローフラグ初期化
            bra     r_RSget
*
* 送信バッファ空 割り込み
*
* 引数    なし
* 返り値  なし
*
reglist             set      d0-d1/a0/a6
TxDataRS:
            ori.w   #$0700,SR
            movem.l reglist,-(sp)
            lea     RsWORK(pc),WREG
            sf      bufffull(WREG)
            move.w  readTx(WREG),d0             * 最新データあり?
            cmp.w   writeTx(WREG),d0
            bne     TxDR000
            move.b  #$28,$00e98005              * 送出せずに終了
            bra     r_TxDataRS
TxDR000:
            lea     Txbuff(WREG),a0
            move.b  (a0,d0.w),$00e98007         * バッファから送出
            addq.w  #1,d0
            andi.w  #256-1,d0
            move.w  d0,readTx(WREG)
r_TxDataRS:
            move.b  #$38,$00e98005              * 割り込みクリア
            movem.l (sp)+,reglist
            rte
*
* 外部ステータス変化 割り込み
*
* 引数    なし
* 返り値  なし
*
ExtStat:
            ori.w   #$0700,SR
            tst.b   $00e98007                   * 1byte 読み飛ばす
            move.b  #$30,$00e98005
            move.b  #$38,$00e98005              * 割り込みクリア
r_ExtStat:
            rte
*
* 1バイト入力 割り込み
*
* 引数    なし
* 返り値  なし
*
reglist             set      d0-d1/a0/a6
RxDataRS:
            ori.w   #$0700,SR
            movem.l reglist,-(sp)
            lea     RsWORK(pc),WREG
            lea     Rxbuff(WREG),a0
            move.w  writeRx(WREG),d1
RxDR000:
            btst.b  #0,$00e98005                * 受信データがあるか?
            beq     r_RxDataRS
            move.b  $00e98007,d0                * データ取得
            cmpi.b  #$FE,d0                     * アクティブセンシングか?
            beq     RxDR000
            move.b  d0,(a0,d1.w)                * バッファにストア
            addq.w  #1,d1
            andi.w  #256-1,d1
            move.w  d1,writeRx(WREG)
            cmp.w   readRx(WREG),d1             * オーバーフローか?
            beq     bufoverRS
            bra     RxDR000
r_RxDataRS:
            move.b  #$38,$00e98005              * 割り込みクリア
            movem.l (sp)+,reglist
            rte
bufoverRS:
            btst.b  #0,$00e98005                * 受信データなしか?
            beq     r_bufoverRS
            move.b  $00e98007,d0                * データ切り捨て
            bra     bufoverRS
r_bufoverRS:
            st      overflow(WREG)              * オーバーフロー発生
            bra     r_RxDataRS
*
* 特別受信条件 割り込み
*
* 引数    なし
* 返り値  なし
*
SpecialRx:
            ori.w   #$0700,SR
            tst.b   $00e98007                   * 1byte 読み飛ばす
            move.b  #$30,$00e98005
            move.b  #$38,$00e98005              * 割り込みクリア
r_SpcialRx:
            rte
            .even
*
* データ領域
*
RSdata:
            dc.b    $09,$80,$04,$44,$01,$00,$03,$c0
            dc.b    $05,$e2,$09,$01,$0b,$50,$0c,$03
            dc.b    $0d,$00,$0e,$02,$03,$c1,$05,$ea
            dc.b    $0e,$03,$10,$10,$01,$12,$09,$09
VecTable:
            dc.l    NULL                 * IRQ0 user call
            dc.l    NULL                 * IRQ1 user call
            dc.l    NULL                 * IRQ2 user call
            dc.l    NULL                 * IRQ4 user call
            dc.l    NULL                 * IRQ7 user call
MIDIvectbl:
            dc.w    MIDIIRQ0-MIDIvectbl
            dc.w    MIDIIRQ1-MIDIvectbl
            dc.w    MIDIIRQ2-MIDIvectbl
            dc.w    MIDIIRQ3-MIDIvectbl
            dc.w    MIDIIRQ4-MIDIvectbl
            dc.w    MIDIIRQ5-MIDIvectbl
            dc.w    MIDIIRQ6-MIDIvectbl
            dc.w    MIDIIRQ7-MIDIvectbl
MIDIvec:
            dc.l    NULL                 * IRQ0 ($80)
            dc.l    NULL                 * IRQ1 ($82)
            dc.l    NULL                 * IRQ2 ($84)
            dc.l    NULL                 * IRQ3 ($86)
            dc.l    NULL                 * IRQ4 ($88)
            dc.l    NULL                 * IRQ5 ($8a)
            dc.l    NULL                 * IRQ6 ($8c)
            dc.l    NULL                 * IRQ7 ($8e)
RSvec:
            dc.l    NULL                 * SCCA ($58)
            dc.l    NULL                 * SCCA ($59)
            dc.l    NULL                 * SCCA ($5a)
            dc.l    NULL                 * SCCA ($5b)
            dc.l    NULL                 * SCCA ($5c)
            dc.l    NULL                 * SCCA ($5d)
            dc.l    NULL                 * SCCA ($5e)
            dc.l    NULL                 * SCCA ($5f)
abort_ssp:          dc.l    0
abort_vec:          dc.l    0
VecWORK:  dcb.b     4,0                  * 割り込み処理用ワークエリア
MidiWORK: dcb.b     522,0                * MIDI処理用ワークエリア
RsWORK:             dcb.b   522,0        * RS232C処理用ワークエリア
            .text
            .offset 0
readTx:             ds.w    1            * 送信 読みだしポインタ
writeTx:  ds.w      1                    * 送信 書き込みポインタ
readRx:             ds.w    1            * 受信 読みだしポインタ
writeRx:  ds.w      1                    * 受信 書き込みポインタ
overflow: ds.b      1                    * オーバーフローフラグ
bufffull: ds.b      1                    * バッファフルフラグ
Txbuff:             ds.b    256          * 送信バッファ
Rxbuff:             ds.b    256          * 受信バッファ
            .text
            .end
```

リスト2

```
* "miditest.s"
* #Source File No. 100.1.1991.10.24
*
* MIDIアクセス・ライブラリを使った
*         テスト・プログラム
*  Copyright (c)1991 Oh!X / Ussy.
*
            .include        iocscall.mac
            .include        doscall.mac

.ifndef             _KEY_INIT
_KEY_INIT equ       $3
.endif
*
* ここからが重要
*
            .xref   Check_MIDI      * MIDI ボード存在チェック
            .xref   Init_MIDI       * MIDI ボード初期化
            .xref   SetUserVec      * IRQ ユーザーコール設定
            .xref   SetVecMIDI      * IRQ ベクタ設定
            .xref   RmvVecMIDI      * IRQ ベクタ解除
            .xref   MIDIput         * MIDI 1byte 出力
            .xref   MIDIget         * MIDI 1byte 入力

            .xref   MIDIrmsg        * Realtime Message 出力
            .xref   MIDIGtimer      * 汎用タイマー・カウンタ設定
            .xref   MIDICtimer      * クロック・カウンタ設定
            .xref   MIDICcounter    * クリック・カウンタ設定
            .xref   MIDIctrlRx      * 受信バッファ制御
            .xref   MIDIIRQctrl     * IRQ1/IRQ4 割り込みモード設定
            .xref   MIDIctrlS       * リアルタイム・メッセージ制御
            .xref   MIDIreadClcok   * リアルタイム・クロック読み出し
            .xref   MIDIsetAHNT     * アドレスハンタの設定

            .xref   Init_RS         * RS ポート初期化
            .xref   SetVecRS        * RS 割り込みベクタ設定
```

▶そろそろスキーのシーズン，初冠雪のニュースがうれしいですね。あとは雪がたくさん降って，たくさん滑れることを祈るだけ。　本橋　正成(21)東京都

```
		.xref	RmvVecRS	* RS 割り込みベクタ解除
		.xref	RSput		* RS 1byte 出力
		.xref	RSget		* RS 1byte 入力
*
* ここまでが重要
*
		.text
main:
		lea	WORK(pc),a6	* ワーク先頭
		lea	$10(a0),a0	* メモリブロックの変更
		lea	WORKSIZE(a1),a1
		suba.l	a0,a1
		pea	(a1)
		pea	(a0)
		DOS	_SETBLOCK
		addq.l	#8,sp
		clr.l	-(sp)		* to SUPER
		DOS	_SUPER
		addq.l	#4,sp
		pea	title(pc)	* タイトルの表示
		DOS	_PRINT
		addq.l	#4,sp
		moveq	#1,d1
		moveq	#2,d2
		IOCS	_TGUSEMD
		IOCS	_OS_CUROF
		bsr	TextClr		* テキスト画面クリア
		move.w	#$FFFF,$e82208	* Text Palet
		move.w	#$E3EA,$e82210
		move.w	#$E3EA,$e82218
		move.b	#1,MIDIch(a6)	* 初期設定
		sf	Voice(a6)
		move.b	#48,Oct(a6)
		sf	Device(a6)
		sf	MIDIflag
		sf	RSflag
		lea	notework(a6),a0
		moveq	#(128/4)-1,d7
M000:
		clr.l	(a0)+
		dbra	d7,M000
		lea	notebuff(a6),a0
		lea	keybuff(a6),a1
		moveq	#16-1,d7
M010:
		st	(a0)+
		sf	(a1)+
		dbra	d7,M010
M019:
		bsr	ChangeDev	* デバイス設定
		bsr	DrawKey		* キーボード表示
		bsr	PutMIDIch	* MIDI ch.表示
		bsr	PutVoice	* Voice 表示
		bsr	PutOct		* Octave 表示
		bsr	PutDevice	* Device 表示
M020:
		bsr	GetKey		* All _BITSNS
		bsr	CheckKey	* キー入力をチェック
		tst.l	d0		* ESC か?
		beq	M020
return:
		bsr	ALLOFF		* ノート情報初期化
		tst.b	Device(a6)
		bne	R010
R000:
		bsr	Check_MIDI	* MIDI board ありか?
		tst.l	d0
		bmi	R020
		bsr	RmvVecMIDI	* IRQ ベクタ解除
		bra	R020
R010:
		bsr	RmvVecRS	* RS 割り込みベクタ解除
R020:
		bsr	TextClr		* テキスト画面クリア
		moveq	#4,d1		* パレット初期化
		moveq	#-2,d2
		IOCS	_TPALET
		moveq	#8,d1
		moveq	#-2,d2
		IOCS	_TPALET
		moveq	#12,d1
		moveq	#-2,d2
		IOCS	_TPALET
		moveq	#1,d1
		moveq	#3,d2
		IOCS	_TGUSEMD
		IOCS	_B_SFTSNS	* キーバッファクリア
		lsr.w	#8,d0
		move.w	d0,d1
		IOCS	_KEY_INIT
		IOCS	_OS_CURON
		pea	goodbye(pc)	* さ～らばいばい
		DOS	_PRINT
		addq.l	#4,sp
		DOS	_EXIT		* Human68k へ
*
* ノート情報のクリア
*
NoteClr:
		lea	notework(a6),a5
		moveq	#(128/4)-1,d7
NC000:
		clr.l	(a5)+
		dbra	d7,NC000
r_NoteClr:
		rts
*
* すべての _BITSNS を行なう
*
GetKey:
		lea	keybuff(a6),a5	* _BITSNS 情報エリア
		moveq	#0,d1
GK000:
		IOCS	_BITSNS
		move.b	d0,(a5)+
		addq.b	#1,d1
		cmpi.b	#16,d1
		bcs	GK000
r_GetKey:
		rts
*
* 特定キーのチェック
*
CheckKey:
		lea	keybuff(a6),a5	* _BITSNS 情報エリア
CK000:
		btst	#1,0(a5)	* ESC
		bne	CK999
		btst	#0,7(a5)	* ROLL UP
		bne	MIDIchUP
		btst	#1,7(a5)	* ROLL DOWN
		bne	MIDIchDOWN
		btst	#2,7(a5)	* UNDO
		bne	DevChange
		btst	#5,7(a5)	* →
		bne	OctUP
		btst	#3,7(a5)	* ←
		bne	OctDOWN
		btst	#4,7(a5)	* ↑
		bne	VoiceUP
		btst	#6,7(a5)	* ↓
		bne	VoiceDOWN
CK010:
		moveq	#0,d7
		lea	KeyWork(pc),a4	* キーチェック表
CK011:
		move.w	d7,d6
		lsl.w	#2,d6
		moveq	#0,d4
		moveq	#0,d5
		move.b	0(a4,d6.w),d4	* key code grp
		move.b	1(a4,d6.w),d5	* key code bit
		move.b	2(a4,d6.w),d6	* note (offset)
		add.b	Oct(a6),d6	* real note
		btst	d5,(a5,d4.w)	* キーオンか?
		beq	CK012
		pea	CK013(pc)	* 戻り場所
		bra	KeyON		* キーオン
CK012:
		bsr	KeyOFF		* キーオフ
CK013:
		addq.w	#1,d7
		cmpi.w	#19,d7		* 19個のキーをチェック
		bcs	CK011
r_CheckKey:
		moveq	#0,d0		* 無事終了
		rts
CK999:
		moveq	#-1,d0		* ESC キーが押された
		rts
MIDIchUP:
		bsr	ALLOFF		* 初期化
		addq.b	#1,MIDIch(a6)
		andi.b	#15,MIDIch(a6)
		bsr	PutMIDIch	* MIDI ch. 表示
		bra	r_CheckKey
MIDIchDOWN:
		bsr	ALLOFF		* 初期化
		subq.b	#1,MIDIch(a6)
		andi.b	#15,MIDIch(a6)
		bsr	PutMIDIch	* MIDI ch. 表示
		bra	r_CheckKey
OctUP:
		bsr	ALLOFF		* 初期化
		move.b	Oct(a6),d0
		cmpi.b	#108,d0
		bcc	r_OctUP
		addi.b	#12,d0
r_OctUP:
		move.b	d0,Oct(a6)
		bsr	PutOct		* Octave 表示
		bra	r_CheckKey
OctDOWN:
		bsr	ALLOFF		* 初期化
		move.b	Oct(a6),d0
		cmpi.b	#12,d0
		bcs	r_OctDOWN
		subi.b	#12,d0
r_OctDOWN:
		move.b	d0,Oct(a6)
		bsr	PutOct		* Octave 表示
		bra	r_CheckKey
VoiceUP:
		bsr	ALLOFF		* 初期化
		addq.b	#1,Voice(a6)
		andi.b	#127,Voice(a6)
		bsr	PutVoice	* Voice 表示
		bra	r_CheckKey
VoiceDOWN:
		bsr	ALLOFF		* 初期化
		subq.b	#1,Voice(a6)
		andi.b	#127,Voice(a6)
		bsr	PutVoice	* Voice 表示
		bra	r_CheckKey
DevChange:
		bsr	ALLOFF		* 初期化
		not.b	Device(a6)
		bsr	ChangeDev	* 制御デバイス変更
		bsr	PutDevice	* デバイス表示
		bra	r_CheckKey
*
* キーONルーチン
*
* 入力	d6.b	ノート番号
* 返り値	なし
*
KeyON:
		lea	notebuff(a6),a3
		moveq	#16-1,d5
KN000:
		cmp.b	(a3)+,d6	* すでに発音している?
		beq	r_KeyON
		dbra	d5,KN000
		lea	notebuff(a6),a3
		moveq	#16-1,d5
KN001:
		tst.b	(a3)+
		bmi	KN010		* 空きバッファ
		dbra	d5,KN001
		bra	r_KeyON		* 空きなし
KN010:
		subq.l	#1,a3
		move.b	d6,(a3)		* ノートをストック
KN011:
		lea	notework(a6),a3
		ext.w	d6
		st	(a3,d6.w)
KN012:
		movea.l	PutProc(a6),a3	* 出力ルーチン
		moveq	#$90,d0		* note on
		add.b	MIDIch(a6),d0
		jsr	(a3)
		move.b	d6,d0		* note
		jsr	(a3)
		moveq	#127,d0		* velocity
		jsr	(a3)
KN020:
		ext.l	d6
		divu	#12,d6
		move.l	d6,d5
		swap	d6
		lsl.w	#2,d6
		move.l	d5,d4
		lsl.w	#3,d4
		sub.w	d5,d4
		lea	$00e6080a,a3
		lea	(a3,d4.w),a3
		lea	NoteON(pc),a2
		move.w	0(a2,d6.w),d5	* Data Offset
		move.w	2(a2,d6.w),d4	* Address Offset
		lea	(a3,d4.w),a3
		lea	(a2,d5.w),a2
		moveq	#48-1,d4
KN030:
		move.b	(a3),d0		* キーマークを表示
		or.b	(a2)+,d0
```

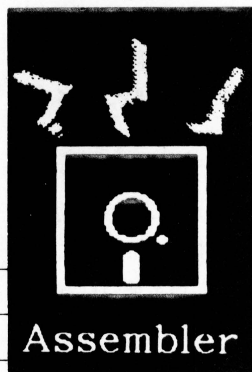
▶11月号の特集はよかったですね。特にスターウォーズの記事が大きく紹介されていたのがうれしいです。記事を読んだらますますほしくなってしまいました。ああ，発売日が待ちどおしい。
伊藤　直樹(23)愛知県

```
        move.b  d0,(a3)
        move.b  1(a3),d0
        or.b    (a2)+,d0
        move.b  d0,1(a3)
        lea     $80(a3),a3
        dbra    d4,KN030
r_KeyON:
        rts
*
*  全キーOFFルーチン
*
ALLOFF:
        lea     notebuff(a6),a3
        moveq   #16-1,d5
AF000:
        moveq   #0,d6               * バッファを初期化
        move.b  (a3),d6
        st      (a3)+
        tst.b   d6
        bmi     AF001
        movem.l d5/a3,-(sp)
        bsr     KF020               * キーマークを消す
        movem.l (sp)+,d5/a3
AF001:
        dbra    d5,AF000
        movea.l PutProc(a6),a3      * 出力ルーチン
        moveq   #15,d6
AF002:
        moveq   #$b0,d0             * 全ノートオフ
        add.b   d6,d0
        jsr     (a3)
        moveq   #$7b,d0
        jsr     (a3)
        moveq   #0,d0
        jsr     (a3)
        moveq   #$b0,d0             * ペダルオフ
        add.b   d6,d0
        jsr     (a3)
        moveq   #$40,d0
        jsr     (a3)
        moveq   #0,d0
        jsr     (a3)
        dbra    d6,AF002            * 16 パート分
AF010:
        bsr     GetKey
        lea     keybuff(a6),a5
        tst.b   7(a5)               * キーが離された?
        bne     AF010
r_ALLOFF:
        rts
*
*  キーOFFルーチン
*
*  入力    d6.b    ノート番号
*  返り値  なし
*
KeyOFF:
        lea     notework(a6),a3
        ext.w   d6
        tst.b   (a3,d6.w)           * キー ON している?
        beq     r_KeyOFF
        sf      (a3,d6.w)
        lea     notebuff(a6),a3
        moveq   #16-1,d5
KF000:
        cmp.b   (a3)+,d6            * ノートを捜す
        beq     KF010
KF001:
        dbra    d5,KF000
        bra     r_KeyOFF            * 該当ノートなし
KF010:
        subq.l  #1,a3
        st      (a3)                * ノートをクリア
KF011:
        movea.l PutProc(a6),a3
        moveq   #$80,d0             * note off
        add.b   MIDIch(a6),d0
        jsr     (a3)
        move.b  d6,d0               * note
        jsr     (a3)
        moveq   #0,d0               * velocity
        jsr     (a3)
KF020:
        ext.l   d6
        divu    #12,d6
        move.l  d6,d5
        swap    d6
        lsl.w   #2,d6
        move.l  d5,d4
        lsl.w   #3,d4
        sub.w   d5,d4
        lea     $00e6080a,a3
        lea     (a3,d4.w),a3
        lea     NoteON(pc),a2
        move.w  0(a2,d6.w),d5       * Data Offset
        move.w  2(a2,d6.w),d4       * Address Offset
        lea     (a3,d4.w),a3
        lea     (a2,d5.w),a2
        moveq   #48-1,d4
KF030:
        move.b  (a3),d0             * キーマークを消す
        move.b  (a2)+,d1
        not.b   d1
        and.b   d1,d0
        move.b  d0,(a3)
        move.b  1(a3),d0
        move.b  (a2)+,d1
        not.b   d1
        and.b   d1,d0
        move.b  d0,1(a3)
        lea     $80(a3),a3
        dbra    d4,KF030
r_KeyOFF:
        rts
*
*  オクターブ表示
*
PutOct:
        lea     $00e62280,a0
        moveq   #(128/4)-1,d7
PO000:
        clr.l   $000(a0)            * 矢印を表示する
        clr.l   $080(a0)            * 部分を消去
        clr.l   $100(a0)
        clr.l   $180(a0)
        clr.l   $200(a0)
        clr.l   $280(a0)
        clr.l   $300(a0)
        clr.l   $380(a0)
        addq.l  #4,a0
        dbra    d7,PO000
        moveq   #0,d6               * 表示座標を求める
        move.b  Oct(a6),d6
        divu    #12,d6
        move.w  d6,-(sp)
        move.w  d6,d5
        lsl.w   #3,d6
        sub.w   d5,d6
        lea     $00e6228a,a0        * 演奏可能な範囲を表示
        lea     (a0,d6.w),a0
        lea     OctArw(pc),a1       * 矢印のデータアドレス
        moveq   #11-1,d7
PO010:
        move.b  (a1)+,$000(a0)
        move.b  (a1)+,$080(a0)
        move.b  (a1)+,$100(a0)
        move.b  (a1)+,$180(a0)
        move.b  (a1)+,$200(a0)
        move.b  (a1)+,$280(a0)
        move.b  (a1)+,$300(a0)
        move.b  (a1)+,$380(a0)
        addq.l  #1,a0
        dbra    d7,PO010
PO020:
        move.w  (sp)+,d6
        lsl.w   #2,d6
        lea     OctNO(pc),a0        * 文字列を表示
        lea     (a0,d6.w),a0
        pea     Octmsg(pc)
        DOS     _PRINT
        addq.l  #4,sp
        pea     (a0)
        DOS     _PRINT
        addq.l  #4,sp
r_PutOct:
        rts
*
*  MIDI Ch. を表示
*
PutMIDIch:
        pea     Mchmsg(pc)
        DOS     _PRINT
        addq.l  #4,sp
        moveq   #0,d0
        move.b  MIDIch(a6),d0
        bsr     PutDEC
r_PutMIDIch:
        rts
*
*  音色番号を表示
*
PutVoice:
        pea     Voimsg(pc)
        DOS     _PRINT
        addq.l  #4,sp
        moveq   #0,d0
        move.b  Voice(a6),d0
        bsr     PutDEC
        movea.l PutProc(a6),a3
        moveq   #$c0,d0             * prog. change
        add.b   MIDIch(a6),d0
        jsr     (a3)
        move.b  Voice(a6),d0        * prog. number
        jsr     (a3)
r_PutVoice:
        rts
*
*  制御しているデバイス名を表示
*
PutDevice:
        pea     control(pc)
        DOS     _PRINT
        addq.l  #4,sp
        tst.b   Device(a6)          * MIDI ? RS-232C
        beq     PV001
PV000:
        pea     rsmsg(pc)
        bra     PV010
PV001:
        tst.b   MIDIflag(a6)        * MIDI board あり?
        beq     PV002
        pea     midimsg(pc)
        bra     PV010
PV002:
        pea     nomidimsg(pc)       * ダメなのだ~
PV010:
        DOS     _PRINT
        addq.l  #4,sp
r_PutDevice:
        rts
*
*  制御デバイスの変更
*
ChangeDev:
        tst.b   Device(a6)          * MIDI or RS set?
        bne     RS000
MIDI000:
        tst.b   RSflag(a6)          * RS 使用中?
        beq     MIDI100
        bsr     RmvVecRS            * RS 割り込みベクタ解除
MIDI100:
        bsr     Check_MIDI          * MIDI board ありか?
        tst.l   d0
        bmi     MIDI999
MIDI110:
        bsr     Init_MIDI           * MIDI ボード初期化
        bsr     SetVecMIDI          * IRQ ベクタ設定
        st      MIDIflag(a6)
        sf      RSflag(a6)
        lea     MIDIput(pc),a0      * MIDI 1byte 出力
        move.l  a0,PutProc(a6)
        bra     r_ChangeDev
MIDI999:
        sf      MIDIflag(a6)
        sf      RSflag(a6)
        lea     noPut(pc),a0        * ダミー出力ルーチン
        move.l  a0,PutProc(a6)
        bra     r_ChangeDev
RS000:
        tst.b   MIDIflag(a6)        * MIDI 使用中?
        beq     RS100
        bsr     RmvVecMIDI          * IRQ ベクタ解除
RS100:
        bsr     Init_RS             * RS ポート初期化
        bsr     SetVecRS            * RS 割り込みベクタ設定
        sf      MIDIflag(a6)
        st      RSflag(a6)
        lea     RSput(pc),a0        * RS 1byte 出力
        move.l  a0,PutProc(a6)
r_ChangeDev:
        rts
*
*  10進数(3桁)の表示
*
PutDEC:
        lea     DEC2STR2(pc),a0
        moveq   #0,d6
PD000:
        moveq   #0,d7
        move.w  (a0)+,d1
        beq     PD005
PD001:
        sub.w   d1,d0
        bcs     PD002
        addq.w  #1,d7
        bra     PD001
```

▶「STUDIO X」にあるタイトル横のカットがかわいくて好きです。誰が描いているのかな。今度，結婚を前提にお付き合いしたいです。なんてね。　加賀　稔(23)愛知県

```
PD002:
                add.w   d1,d0
PD003:
                addi.w  #'0',d7
PD004:
                move.l  d0,-(sp)
                move.w  d7,-(sp)
                DOS     _PUTCHAR
                addq.l  #2,sp
                move.l  (sp)+,d0
                bra     PD000
PD005:
                rts
*
* 裏テキスト画面のクリア
*
TextClr:
                move.w  #$01c0,$00e8002a
                lea     $00e00000,a1
                move.l  #$4000-1,d7
TC000:
                clr.l   (a1)+
                clr.l   (a1)+
                dbra    d7,TC000
                move.w  #$0033,$00e8002a
r_TextClr:
                rts
*
* ダミー 1 byte 出力ルーチン
*
noPut:
                rts
*
* キーボード表示ルーチン
*
DrawKey:
                lea     $00e4080a,a0
                lea     KeyTBL(pc),a1
                moveq   #77-1,d7
                moveq   #0,d6
DK000:
                move.w  d6,d1
                add.w   d1,d1
                move.w  (a1,d1.w),d1
                lea     (a1,d1.w),a2
                move.l  a0,-(sp)
                moveq   #48-1,d5
DK010:
                move.b  (a2)+,(a0)
                lea     $80(a0),a0
                dbra    d5,DK010
                movea.l (sp)+,a0
                addq.l  #1,a0
                addq.b  #1,d6
                cmpi.b  #7,d6
                bcs     DK020
                sf      d6
DK020:
                dbra    d7,DK000
r_DrawKey:
                rts
*
* データエリア
*
                .data
DEC2STR2:
                dc.w    100,10,1,0              * 10進数表示用
KeyTBL:
                dc.w    NoteC-KeyTBL            * 鍵盤データテーブル
                dc.w    NoteD-KeyTBL
                dc.w    NoteE-KeyTBL
                dc.w    NoteF-KeyTBL
                dc.w    NoteG-KeyTBL
                dc.w    NoteA-KeyTBL
                dc.w    NoteB-KeyTBL
NoteC:
                dcb.b   31,%1111_0000           * 鍵盤データ
                dc.b       %1111_1000
                dcb.b   15,%1111_1110
                dc.b       %0111_1110
NoteD:
                dcb.b   31,%0011_1000
                dc.b       %0111_1100
                dcb.b   15,%1111_1110
                dc.b       %0111_1100
NoteE:
                dcb.b   31,%0001_1110
                dc.b       %0011_1110
                dcb.b   15,%1111_1110
                dc.b       %1111_1100
NoteF:
                dcb.b   31,%1111_0000
                dc.b       %1111_1000
                dcb.b   15,%1111_1110
                dc.b       %0111_1110
NoteG:
                dcb.b   31,%0011_1000
                dc.b       %0111_1100
                dcb.b   15,%1111_1110
                dc.b       %0111_1100
NoteA:
                dcb.b   31,%0001_1100
                dc.b       %0011_1110
                dcb.b   15,%1111_1110
                dc.b       %0111_1100
NoteB:
                dcb.b   31,%0000_1110
                dc.b       %0001_1110
                dcb.b   15,%1111_1110
                dc.b       %0111_1100
                .even
KeyWork:                                        * キースキャン用データ
                dc.b    5,2,0,0                 * Z
                dc.b    3,7,1,0                 * S
                dc.b    5,3,2,0                 * X
                dc.b    4,0,3,0                 * D
                dc.b    5,4,4,0                 * C
                dc.b    5,5,5,0                 * V
                dc.b    4,2,6,0                 * G
                dc.b    5,6,7,0                 * B
                dc.b    4,3,8,0                 * H
                dc.b    5,7,9,0                 * N
                dc.b    4,4,10,0                * J
                dc.b    6,0,11,0                * M
                dc.b    6,1,12,0                * ,
                dc.b    4,6,13,0                * L
                dc.b    6,2,14,0                * .
                dc.b    4,7,15,0                * ;
                dc.b    6,3,16,0                * /
                dc.b    6,4,17,0                * _
                dc.b    5,1,18,0                * ]
NoteON:
                dc.w    onC-NoteON,0            * キーマーク表示データ
                dc.w    onC_-NoteON,0           * のテーブル
                dc.w    onD-NoteON,1
                dc.w    onD_-NoteON,1
                dc.w    onE-NoteON,2
                dc.w    onF-NoteON,3
                dc.w    onF_-NoteON,3
                dc.w    onG-NoteON,4
                dc.w    onG_-NoteON,4
                dc.w    onA-NoteON,5
                dc.w    onA_-NoteON,5
                dc.w    onB-NoteON,6
*
* キーマークデータ
*   描画&消去用のマスクも兼ねる
*
onC:
                dcb.w   31,%11110000_00000000
                dc.w       %11111000_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111110_00000000
onC_:
                dcb.w   31,%00000111_10000000
                dcb.w   17,%00000000_00000000
onD:
                dcb.w   31,%00111000_00000000
                dc.w       %01111100_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111100_00000000
onD_:
                dcb.w   31,%00000011_11000000
                dcb.w   17,%00000000_00000000
onE:
                dcb.w   31,%00011110_00000000
                dc.w       %00111110_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %11111100_00000000
onF:
                dcb.w   31,%11110000_00000000
                dc.w       %11111000_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111110_00000000
onF_:
                dcb.w   31,%00000111_10000000
                dcb.w   17,%00000000_00000000
onG:
                dcb.w   31,%00111000_00000000
                dc.w       %01111100_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111100_00000000
onG_:
                dcb.w   31,%00000011_11000000
                dcb.w   17,%00000000_00000000
onA:
                dcb.w   31,%00011100_00000000
                dc.w       %00111110_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111100_00000000
onA_:
                dcb.w   31,%00000001_11100000
                dcb.w   17,%00000000_00000000
onB:
                dcb.w   31,%00001110_00000000
                dc.w       %00011110_00000000
                dcb.w   15,%11111110_00000000
                dc.w       %01111100_00000000
                .even
*
* オクターブ範囲を表示する矢印のデータ
*
OctArw:
                dc.b    $10,$38,$7c,$fe,$38,$38,$3f,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $00,$00,$00,$00,$00,$00,$ff,$00
                dc.b    $08,$1c,$3e,$7f,$1c,$1c,$fc,$00
                .even
title:
                dc.b    $1b,'[2J',$1b,'[37m',9,9,'MIDI Access'
                dc.b    ' Subroutine test program  Copyright '
                dc.b    '(c)1991 Oh!X / Ussy.'
                dc.b    $1b,'[m',13,10,0
goodbye:
                dc.b    $1b,'[2J',$1b,'[37mMIDI Access'
                dc.b    ' Subroutine test program  Copyright '
                dc.b    '(c)1991 Oh!X / Ussy.',13,10
                dc.b    $1b,'[36m MIDI(CZ-6BM1/SX-68M), '
                dc.b    'RS-232C による入出力'
                dc.b    'テストを行ないました。'
                dc.b    $1b,'[m',13,10,0
control:
                dc.b    $1b,'[6;0H',9,9,9,$1b,'[37mControl Device :
                dc.b    $1b,'[0K',0
midimsg:
                dc.b    $1b,'[35mMIDI(CZ-6BM1/SX-68M)',$1b,'[m',0
rsmsg:
                dc.b    $1b,'[36mRS-232C(Internal)',$1b,'[m',0
nomidimsg:
                dc.b    $1b,'[35mMIDIボードがありません',$1b,'[m',0
Mchmsg:
                dc.b    $1b,'[2;0H',$1b,'[35mMIDIch:',0
Voimsg:
                dc.b    $1b,'[3;0H',$1b,'[37mVoice :',0
Octmsg:
                dc.b    $1b,'[4;0H',$1b,'[36mOctave:',0
OctNO:
                dc.b    ' -1',0                 * オクターブ表示用データ
                dc.b    '  0',0
                dc.b    '  1',0
                dc.b    '  2',0
                dc.b    '  3',0
                dc.b    '  4',0
                dc.b    '  5',0
                dc.b    '  6',0
                dc.b    '  7',0
                dc.b    '  8',0
                dc.b    '  9',0
                .even
WORK:
                .text
                .offset 0
PutProc:  ds.l    1                             * 1 byte Put Proc Address
MIDIflag: ds.b    1                             * MIDI on flag
RSflag:           ds.b    1                     * RS on flag
notework: ds.b    128                           * Note ON/OFF Check
keybuff:  ds.b    16                            * _BITSNS work area
notebuff: ds.b    16                            * Note ON key buffer
MIDIch:           ds.b    1                     * MIDI ch.
Voice:            ds.b    1                     * Voice work
Oct:              ds.b    1                     * Current Octave
Device:           ds.b    1                     * Control Device
WORKSIZE
                .end
*
* これにて一件落着なり
*
```

自由変形ルーチンの作成



Murata Toshiyuki 村田 敏幸

グラフィック関係のお話も今回でひと区切り。そこで，今回は任意の図形をいろいろ変化させる自由変形のお話です。これで一応ひとおりの図形が描けるようになるはずです。回転ルーチンの代用品にもなることだし，積極的に覚えてもらいたいテクニックです。

今月は画像の自由変形を取り上げる。矩形のグラフィックパターンを引き伸ばしたり，押し潰したり，歪めたり，回転したり，ねじったり，裏返したりして，任意の四角形にはめ込んで描く，多目的プットルーチンを作成してみよう。

基本

図1に自由変形の基本的な考え方を示す。要は単純な比の問題だ。元画像上の1点は，変形後の四角形の向かい合った辺どうしを同じ比で分割して，結んだ交点に変換される。実際のプログラムでは，1点ずつ座標変換するのは効率が悪いので，点単位ではなく線分単位で考える。図2のように，描画先を斜めの線分に分割し，水平に1ライン分ずつ切り出した元画像を適当に引き伸ばしたり，縮めたりして，張り付ける感じだ。

この方法で隙間なく描画するためには，描画先の四角形をAB，DCのどちらか長いほうのピクセル数本の線分に分割する必要があり，そうすると，短いほうの辺近くで線分同士が重なる。本来，このような場合は，重なる部分については色の混ぜ合わせを行うべきだが，今回はそこまでやるつもりはない。あとで示すプログラムでは，処理順序にしたがって何も考えずに上書きする。

では，アルゴリズムをつぎの3ステップに分けて考え，個別に具体化していこう。

1）描画先となる線分P_S-P_Eを決める

2）P_S-P_Eに対応する元画像上の水平線分$P_S'-P_E'$を求める

3）$P_S'-P_E'$をP_S-P_Eに張り付ける

●描画先線分の決定

描画先P_S-P_EはADからBCへ傾きを変えながら移動していく。両端点P_SとP_Eに注目するなら，P_SはAB上を，P_EはDC上を移動していくことになり，その移動経路は線分発生アルゴリズムで順次求まる。当然，両端点が決まれば，それらを結ぶ線分もまた決定する。

ここで，ABとDCの長さは必ずしも等しいわけではないから，AB，DC上の各点を1対1に結んだのでは長いほうが余る。正しくは，P_SとP_Eが同時にB，Cに達するよう，短いほうの線分をなぞる速度を適当に遅らせなければならない。この速度調節に

図1

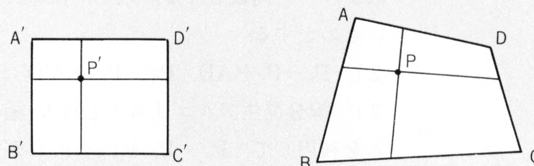


図2

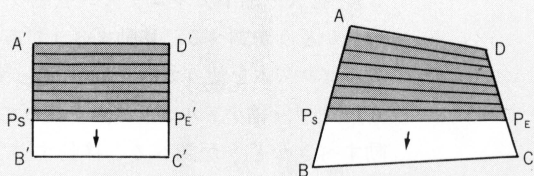
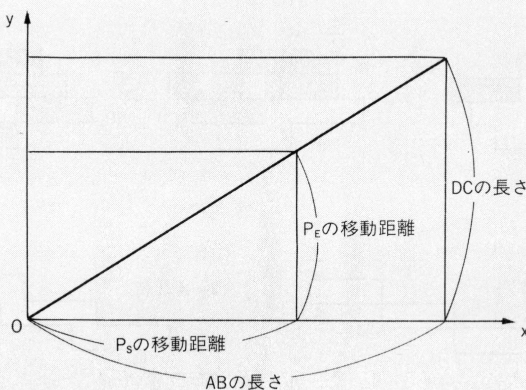


図3



は，拡大・縮小アルゴリズム（その実体は線分発生アルゴリズムと同じもの）が利用できる。仮にABのほうがDCよりも長いとすると，図3のような線分を描くつもりになって，x座標を更新するタイミングでP_Sを，y座標を更新するタイミングでP_Eを移動すればよいわけだ。

●描画先線分に対応する元画像上の水平線分の決定

$P_S'-P_E'$は，P_S-P_EがADからBCに移動するのと連動してA'D'からB'C'へと動いていく。したがって，適当な移動速度で$P_S'-P_E'$を動かしてやれば，P_S-P_Eからわざわざ逆算しなくても，対応する$P_S'-P_E'$が順次求まる。移動速度の調節には，上と同様，拡大・縮小アルゴリズムが応用できる。

●元画像の水平線分を傾けて張り付ける

$P_S'-P_E'$をP_S-P_Eに張り付ける処理は，線分発生アルゴリズムと拡大・縮小アルゴリズムの組み合わせで実現できる。P_S-P_E上の点を順次求めつつ拡大・縮小の手順で対応する$P_S'-P_E'$上の位置を求め，そこから色を拾って点を打っていけばよい。

＊

さて，以上をまとめると，自由変形のアルゴリズムはつぎのようになる。

1）ABとCDの長いほうを選び出す。そのピクセル数がループ回数となる。以下，仮にABのほうが長いものとする

2）P_S-P_EをAD，$P_S'-P_E'$をA'D'に初期化する

3）線分発生アルゴリズムと拡大・縮小アルゴリズムを利用して，$P_S'-P_E'$をP_S-P_Eに張り付ける

4）線分発生アルゴリズムを使ってP_SをABに沿って移動する

5）拡大・縮小アルゴリズムを使ってP_Eを移動すべきかどうか調べる。移動すべきであれば線分発生アルゴリズムを使ってP_EをDCに沿って移動する

6）拡大・縮小アルゴリズムを使って$P_S'-P_E'$を移動すべきかどうか調べる。移動すべきであれば，$P_S'-P_E'$を1ライン下に移動する

7）2）～6）を1）で求めた回数だけ繰り返す

図4

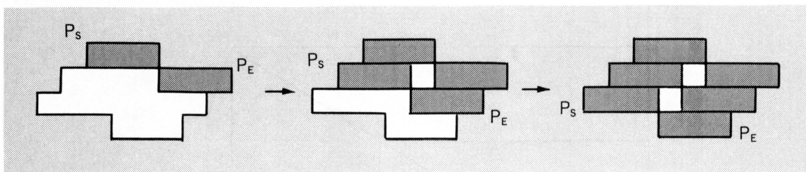
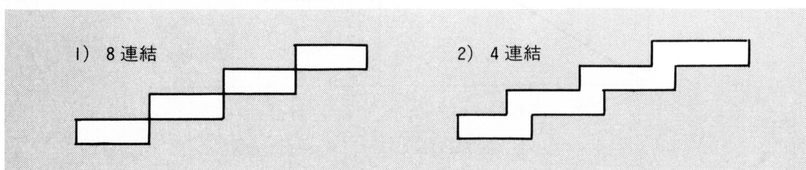


図5



プログラム化のための補足

さて，左で導いたアルゴリズムを実際にプログラムにするうえでは，若干の小細工が必要だ。グラフィック画面上に斜めの線分を並べて面にしようとすると，階段状になった線分の段の部分に隙間を生じる場合がある（図4）。この問題はP_S，P_Eの移動経路に冗長性を持たせることで解決する。具体的には，P_S，P_Eの移動経路を求めるのに，通常使われる8連結（ピクセルが斜めに並ぶことを許す）の線分発生アルゴリズムではなく4連結（ピクセルが必ず縦か横に並ぶ）のアルゴリズムを利用する（図5）。こうすると，ちょうど隙間を埋める（隙間の上を通る）新たな線分が生まれることを確認しておいてほしい。副作用で，同一ピクセルに重複して描画する場面が増えることにはなるが，背に腹は代えられない。

で，その4連結の線分発生アルゴリズムは，8連結のアルゴリズムを多少修正することで簡単に得られる。8連結のアルゴリズムではx，y座標を同時に更新して斜めに移動していた部分を，縦横の2回に分けて移動するようにすればよい。参考までに，リスト1に4連結ラインルーチンのX-BASIC版，ついでに，リスト2にマシン語版を示しておく。以前の8連結版と見比べてみてほしい。

リスト1，2はともに同じアルゴリズムをプログラムにしたものだが，効率上，マシン語版では誤差項を更新するタイミングをX-BASIC版と変えてある。X-BASIC版では誤差項の符号に応じてxを更新するか，yを更新するかを選択するという正直な作りだが，マシン語版では毎回x座標を仮に進め，それに合わせて誤差項も更新してから，誤差項の符号を調べるようになっている。その結果，実はx座標ではなくy座標を更新するタイミングだということがわかったら，進めすぎた分を逆に修正するのだ。

リスト2では，y座標を進めたときの誤差項とG-RAMアドレスそれぞれの変化量にあらかじめ修正分を加味している（62～63行）ことに注目しよう。

プログラムの作成

リスト3に自由変形機能つきのグラフィックパターンプットルーチンを示す。例によってサブルーチンだけなので，動作試験時にはリスト4に用意したメインルーチンを利用してもらいたい。

リスト4は画像を画面に読み込んでおいてから起動すると，その左上128ピクセル四方を切り出し，何かキーが押されるまでランダムな四角形に変形して描画し続ける。

サブルーチンgtputには26～38行のような構造で引数を渡す。描画先は4頂点を反時計回りに並べることになっている。並べる順序にも意味があり，頂点を逆順に並べれば，画像を裏返して描くことができる。これまでとくに触れなかったが，描画先の四角形はへこんでいたり，ねじれていたりしてもかまわない。

リスト3は比較的長いプログラムだが，ただひたすらに，Bresenhamの線分発生アルゴリズムを組み合わせているだけで，あまり見るべき点がない。ここでは，全体の処理の流れを追いながら，ポイントになる部分を拾うにとどめたいと思う。

●初期化（74～143行，291～339行）

初期化部分では，

P_S，P_Eの移動経路決定

P_S，P_Eどちらかの移動速度調節

$P_S'-P_E'$の移動速度調節

のそれぞれに対して，Bresenhamの線分発生アルゴリズムを適用するための誤差項関連の初期化をしている。

76～85行がP_S，P_Eの移動経路関係の初期化だ。実際の誤差項周りの計算は291～339行サブルーチンinitにまとめてある。P_S, P_Eの移動経路は4連結の線分発生アルゴリズムで求めることに注意してほしい。さすがに扱う変数の数が多いので，求めたパラメータは40～52行のような構造のワークにしまっておくようになっている。

ここで，線分発生時には，線分の傾きに応じて処理を振り分けなければならないことを思い出そう。一気に線分全体を発生する場合は，ループに入る直前に1回処理を振り分けるだけですむが，いまの場合，ループの中で2点を平行して発生しなければならない。常識的なやり方では，線分の傾きが45°より

リスト1

```
10 int x0 = 0, y0 = 0
20 int x1 = 5, y1 = 3
30 int e, x, y
40 int dx, dy, sx, sy, i
50 /*
60 screen 2,0,1,1
70 dx = abs(x1-x0): dy = abs(y1-y0)
80 sx = sgn(x1-x0): sy = sgn(y1-y0)
90 x  = x0         : y  = y0
100 if (dx >= dy) then {
110     e = 2*dy - dx
120     for i = 0 to dx + dy
130         pset(x, y, 15)
140         if (e >= 0) then {
150             y = y + sy
160             e = e - 2*dx
170         } else {
180             x = x + sx
190             e = e + 2*dy
200         }
210     next
220 } else {
230     e = 2*dx - dy
240     for i = 0 to dx + dy
250         pset(x, y, 15)
260         if (e >= 0) then {
270             x = x + sx
280             e = e - 2*dx
290         } else {
300             y = y + sy
310             e = e + 2*dx
320         }
330     next
340 }
```

リスト2　gline4. s

```
*           線分描画（4連結）

            .include        gconst.h
            .include        gmacro.h
*
            .xdef   gline4
            .xref   gramadr
            .xref   glclip
*
            .offset 0
*
X0:         .ds.w   1
Y0:         .ds.w   1
X1:         .ds.w   1
Y1:         .ds.w   1
COL:        .ds.w   1
*
            .text
            .even
*
gline4:
ARGPTR      =       8
            link    a6,#0
            movem.l d0-d6/a0-a1,-(sp)

            movea.l ARGPTR(a6),a1   *a1=引数列
            movem.w (a1),d0-d3      *d0～d3に座標を取り出す

            jsr     glclip          *クリッピングする
            bne     done            *Z=0なら完全にウィンドウ
外

            jsr     gramadr         *a0=始点のG-RAMアドレス

            sub.w   d0,d2           *d2=x1-x0
            move.w  d2,d4           *d4=d2
            ABS     d2              *d2=dx=abs(x1-x0)
            SGN     d4              *d4=sx=sgn(x1-x0)

            sub.w   d1,d3           *d3=y1-y0
            move.w  d3,d5           *d5=d3
            ABS     d3              *d3=dy=abs(y1-y0)
            SGN     d5              *d5=sy=sgn(y1-y0)

            add.w   d4,d4           *d4=sx*2
            moveq.l #GSFTCTR,d0     *d5=sy*1024
            asl.w   d0,d5           *  (or sy*2048)

            move.w  COL(a1),d0      *d0=描画色

            move.w  d2,d6           *d6=dx
            add.w   d3,d6           *d6=dx+dy
                                    *  =ループカウンタ

            cmp.w   d3,d2           *dy>dxならば
            bcs     yline           *  yについてループ

                            *dx≧dyのとき
xline:      move.w  d2,d1           *d1=e=-dx
            neg.w   d1              *
            add.w   d2,d2           *d2=2*dx
            add.w   d3,d3           *d3=2*dy
            add.w   d3,d2           *d2=2*dx+2*dy
            sub.w   d4,d5           *d5=yを更新したときの
                                    *  G-RAMアドレス変化量
                                    *(xを先に更新した分の補
正を含む)
                                    *do {
xline0:     move.w  d0,(a0)         *  pset(x, y)
            adda.w  d4,a0           *  x+=sx
            add.w   d3,d1           *  e+=2*dy
            bmi     xline1          *  if (e >= 0) {
            adda.w  d5,a0           *     y+=sy, x-=sx
            sub.w   d2,d1           *     e-=2*dx+2*dy
                                    *  }
xline1:     dbra    d6,xline0       *} while (--n >= 0)
            bra     done

                            *dx<dyのとき
yline:      move.w  d3,d1
            neg.w   d1
            add.w   d2,d2
            add.w   d3,d3
            add.w   d2,d3
            sub.w   d5,d4

yline0:     move.w  d0,(a0)
            adda.w  d5,a0
            add.w   d2,d1
            bmi     yline1
            adda.w  d4,a0
            sub.w   d3,d1

yline1:     dbra'   d6,yline0
done:       movem.l (sp)+,d0-d6/a0-a1
            unlk    a6
            rts

            .end
```

急かなだらかかを覚えておき，それをスイッチにしてループ中で毎回処理を振り分けることになる。が，リスト3では345～358行，364～377行に線分を1ピクセルずつ発生するサブルーチンを傾き別に2つ用意しておき，初期化時点で，その先頭アドレスをアドレスレジスタに入れておくことで，ループ内部での条件分岐を省略できるようにしている。

87～102行ではAB，DCのどちらが長いかを調べ，長いほうのピクセル数をループカウンタとして使うためにd7に格納している。ここで，P_S，P_Eの移動速度は，移動経路の長いほうを基準にして，短いほうを適当に遅らせることで調節するのだから，ABのほうが長い場合はP_S，DCのほうが長いときはP_E，等しいときは両方の速度調節が不要になる。87～102行ではついでにそのあたりのつじつま合わせを行っている。345～358行，364～377行の線分を1ピクセルずつ発生するサブルーチンの先頭についている速度調節処理の部分が実行されないよう，アドレスレジスタに入れたサブルーチンのアドレスを必要なだけ進めているのだ。

初期化部分の残りはあまり面白くない。104～120行は，P_S，P_E(のどちらか)，$P_S'-P_E'$の移動速度調節関係の初期化で，見てのとおり。124～127行もパターンの横幅からパターン1ライン分のバイト数を計算しているだけだ。129～134行ではパターンを画面に張り付けるときの拡大・縮小に使う値のうち，定数となるものを先に計算している。最後に136～143行でP_S，P_Eの座標と，移動経路決定用の誤差項をレジスタに取り出して初期化は終わりだ。

●メインループ（145～184行）

プログラムの実行に必要なパラメータはほとんど初期化時点で計算してあるし，パターン1ライン分を張り付ける処理はサブルーチンにして抜き出してあるので，メインループはかなり簡単になっている。まず，158～165行で1ライン分描画し，167～168行でP_E，170～175行でP_S，177～182行で$P_S'-P_E'$を速度を考慮して移動するという，アルゴリズムどおりの作りだ。

1点，$P_S'-P_E'$の移動速度を調節する部分が，以前作った拡大・縮小ルーチンとはちょっと違っていることを指摘しておこう。誤差項eの符号による処理の振り分けが，

$e<0$ と $e\geqq 0$

ではなく，

$e\leqq 0$ と $e>0$

になっている（178，182行）。前の拡大・縮小ルーチンgxputには“描画先の高さや幅が1ピクセルしかないとうまく動かない”という制限があったが，このようにe＝0の場合の扱いを変更すれば，あんな間抜けな制限はつけなくてもすんだのだ。gxputの該当箇所も修正しておくとよいだろう（回転ルーチンgrputにも同様の部分がある）。

●1ライン分の描画（193～286行）

サブルーチンput1lineでは，線分P_S-P_Eを発生する線分描画アルゴリズムと，$P_S'-P_E'$をP_S-P_Eの長さに揃える拡大・縮小を組み合わせて，1ライン分のパターンを画面に張り付ける。

このサブルーチンではとくにクリッピング部分に触れておきたい。クリッピングは，10～23行で定義した1点分描画するマクロPSETの中で1点ごとに行っている。線分のクリッピングは簡単に実現できるのに，あえて1点ずつクリッピングしているのは，クリッピングによる誤差を避けるためだ。線分のクリッピングは整数化して計算する以上，どんなに正確に計算しても小数点以下の誤差が生じる。斜めの線分を並べて面にするうえでは，そのわずかな誤差が隙間の原因となる。しかなく，1点ごとにクリッピングウィンドウに収まっているかどうか調べるという原始的な方法をとっているのだ。

しかし，この方法にもまた問題がある。クリッピングされていない（－32768～＋32767の座標値をとる）線分に対してBresenhamの線分発生アルゴリズムを適用すると，誤差項の計算が16ビットでは収まらない。リスト3はこの点には目をつむって16ビット計算でごまかしているため，極端に大きく拡大するとオーバーフローを起こす。気になるようなら，誤差項の計算を32ビットで行うように修正してみてほしい。

変形ルーチンの応用

自由変形ルーチンは結構応用範囲が広い。拡大・縮小ルーチンとしても，回転ルーチンとしても使えるし（当然，項点の回転計算はべつにすることになるが），もっと単純に画像の上下左右反転にも利用できる。欠点は実行速度で，さすがに専用ルーチンには及ばない。ただ，回転ルーチンの代用品としては積極的に使ってみてもよいだろう。以前作った回転ルーチンgrputは元画像を斜めに切り出して水平に張り付けるという方法をとったために，描画範囲はあくまで画面に平行な矩形だった。しかし，今回の自由変形ルーチンgtputを使えば，描画範囲自体を回転した結果を得ることができる。また，拡大を伴う回転の場合，gtputのほうがより自然な回転結果を得られる。grputでは回転してから拡大する形になるので，描画先画面の水平・垂直方向にモザイク模様ができるが，gtputではモザイクごと回転する格好になる。逆に，縮小を伴う回転の場合，gtputでは同一ピ

クセルに重複して描画する分，画質が落ちると考えられる。

自由変形の，より一般的な応用としては3D分野がある。3D→2D変換した面に対して画像を張り付けることで，テクスチャマッピングらしい処理が実現できるし，少々強引だが，グラデーションパターンを張り付けて陰影をつければ，グローシェーディング[1)]もどきにもなる。

で，リスト5は後者の例として面白半分に作ってみたグラデーションパターン作成プログラムだ。リスト5のサブルーチンgradpat1は引数で与えられた2色を補間して128×1のパターンを，gradpat2は4色を補間して128×128のパターンを作成する。リスト6はこのgradpat2で作成したパターンを張り付けることで，グラデーションをつけて四角形を描く簡単なデモとなっている。元パターンが128×128しかないので，ある程度以上描画範囲が広くなると拡大特有のギザギザが目立つようになるし，マッハバンド[2)]対策はなにもしていないからあまり質はよくないが，あり合わせにしてはそれらしい結果が得られるようだ。

リスト5については，とくに問題になる点はないと思う。色の補間も座標の補間同様の考え方で実現できる。2色の補間であれば，RGBごとにBresenhamの線分発生アルゴリズムを利用して，色成分を少しずつ変化させればよい。4色の2次元の補間は，自由変形のように縦横の補間を組み合わせることになる。gradpat2では，4色中の2色ずつを補間して中間的なパターンを2つ作り，そのパターン間をまた補間して最終的なパターンを作成している。多少なりとも精度を稼ぎたかったので，中間パターンはRGBの階調を必要以上に細かくして計算するようにしてみた。ちなみに，処理を共通にした都合でgradpat1も一旦高精度で計算してから32階調に落としているが，これはまったくの二度手間で，最終的な結果には影響していない。

さて，実際に試してほしいのだが，リスト5は比較的わずかな改造でグラデーションボックスフィルルーチンに化ける。メインメモリ上にパターンを作成しようが，G-RAMに書き込もうが，プログラムの作り方はたいして変わらないのだ。その改造の際には，内部では多階調で計算していることを利用して，マッハバンドを目立たなくするよう細工することも考えてみるとよい。つまり，輝度が変化する部分に故意に誤差を持ち込んで，輝度変化を目立たなくするわけだ。多階調から32階調に落とす際に，いきなり割算せずに，小数点以下の乱数を加えてから割算するだけでもそれなりの効果がある。より真面目な解法は自由課題としよう。

＊

1年余りにわたってグラフィック描画を取り扱ってきたわけだが，今回でいちおうひと区切りだ。次回からしばらくの間は，基本に戻って，入門っぽいことをやってみる。

1) グローシェーディング：曲面を平面で近似して3D計算してから頂点間の輝度を線形補間することで陰影をつけて曲面らしく見せる技法。

2) マッハバンド：輝度の変化を実際に以上に大きく感じてしまう目の性質から生じる縞模様。

リスト3　gtput. s

```
*               自由変形プットルーチン

                .include        gconst.h
                .include        gmacro.h
*
                .xdef   gtput
                .xref   gramadr
                .xref   cliprect
*
PSET            macro   X,Y,COL,ADR     *クリッピングしつつ
                local   skip            *  点を打つマクロ
                lea.l   cliprect,a5     *
                cmp.w   (a5)+,X         *
                blt     skip            *
                cmp.w   (a5)+,Y         *
                blt     skip            *
                cmp.w   (a5)+,X         *
                bgt     skip            *
                cmp.w   (a5)+,Y         *
                bgt     skip            *
                move.w  COL,ADR         *
skip:                                   *
                endm                    *

*
                .offset 0               *gtputの引数構造
*
X0:             .ds.w   1               *描画先座標       A
Y0:             .ds.w   1               *
X1:             .ds.w   1               *                      B
Y1:             .ds.w   1               *
X2:             .ds.w   1               *                      C
Y2:             .ds.w   1               *
X3:             .ds.w   1               *                      D
Y3:             .ds.w   1               *
PAT:            .ds.l   1               *パターンアドレス
XL:             .ds.w   1               *パターンの横の長さ-1
YL:             .ds.w   1               *パターンの縦の長さ-1
*
                .offset 0               *線分発生用ワーク
*
X:              .ds.w   1               *x座標
Y:              .ds.w   1               *y座標
DX:             .ds.w   1               *y更新時にeから引く定数
DY:             .ds.w   1               *x更新時にeに加える定数
SX:             .ds.w   1               *x増加方向
SY:             .ds.w   1               *y増加方向
E:              .ds.w   1               *誤差項e
LEN:            .ds.w   1               *線分のピクセル数-1
SDX:            .ds.w   1               *速度調節用e補正値
SDY:            .ds.w   1               *速度調節用e増分
EDG:
*
                .offset -EDG*2-2*4              *スタックフレーム
*
WORKTOP:
ST:             .ds.b   EDG             *始点関連ワーク
ED:             .ds.b   EDG             *終点関連ワーク
PHE:            .ds.w   1               *水平スケーリング用e初期値
PHDY:           .ds.w   1               *水平スケーリング用e増分
PVDX:           .ds.w   1               *垂直スケーリング用e補正値
PVDY:           .ds.w   1               *垂直スケーリング用e増分
_A6:            .ds.l   1               *0
_SP:            .ds.l   1               *4
ARGPTR:         .ds.l   1               *8
*
                .text
                .even
*
gtput:
                link    a6,#WORKTOP
                movem.l d0-d7/a0-a5,-(sp)

                movea.l ARGPTR(a6),a5   *a5=引数列

                movem.w X0(a5),d0-d3    *ABを発生するための
                lea.l   ST(a6),a0       *(=始点の経路を決めるための)
                bsr     init            *  誤差項周りの初期化
                movea.l a1,a3           *a3=始点を移動するサブルーチン

                movem.w X3(a5),d0-d1    *DCを発生するための
                movem.w X2(a5),d2-d3    *(=終点の経路を決めるための)
                lea.l   ED(a6),a0       *  誤差項周りの初期化
                bsr     init            *
                movea.l a1,a4           *a4=終点を移動するサブルーチン

```

```
                move.w  ST+LEN(a6),d7       *d7=ABの長さ-1
                move.w  ED+LEN(a6),d0       *d0=DCの長さ-1
                lea.l   xline-xlinex(a3),a3
                                            *ABの方が長いと仮定し
                                            *  仮に始点の速度調節を殺す

                cmp.w   d0,d7               *ABとDCの長さが等しければ
                beq     equskp              *  始点,終点ともに速度調節不要
                bcc     maxskp              *ABの方が長ければ
                                            *  始点については速度調節不要
                                            *DCの方が長ければ
                                            *  終点については速度調節不要
                move.w  d0,d7               *d7=DCの長さ-1
                lea.l   xlinex-xline(a3),a3     *始点の速度調節を復活する

equskp:         lea.l   xline-xlinex(a4),a4     *終点の速度調節を殺す

maxskp:         move.w  d7,d5               *d7=d5=ABとDCの長さの長い方
                neg.w   d5                  *d5=始点（または終点）の
                                            *  移動速度調節用e

                move.w  d7,d0               *
                add.w   d0,d0               *
                move.w  d0,ST+SDX(a6)       *始点移動速度調節用e補正値
                move.w  d0,ED+SDX(a6)       *終点移動速度調節用e補正値
                move.w  d0,PVDX(a6)         *パターンを垂直になぞる速度の
                                            *  調節用e補正値

                move.w  YL(a5),d0           *d0=パターン高-1
                move.w  d0,d6               *d6=パターンを垂直になぞる速度の
                neg.w   d6                  *  調節用e
                add.w   d0,d0               *パターンを垂直になぞる速度の
                                            *  調節用e増分
                move.w  d0,PVDY(a6)         *

                movea.l PAT(a5),a1          *a1=パターン先頭アドレス

                move.w  XL(a5),d0           *d0=パターン幅-1
                movea.w d0,a5               *a5=パターン1ライン分バイト数
                addq.l  #1,a5               *
                adda.l  a5,a5               *

                move.w  d0,d1               *パターンを水平になぞる速度の
                add.w   d1,d1               *  調節用e増分
                move.w  d1,PHDY(a6)         *

                neg.w   d0                  *パターンを水平になぞる速度の
                move.w  d0,PHE(a6)          *  調節用e

                movem.w ED+X(a6),d2-d3      *(d2.w,d3.w)=終点座標
                move.w  ED+E(a6),d4         *d4.w=終点移動経路決定用e
                swap.w  d2                  *それぞれレジスタの
                swap.w  d3                  *  上位ワードに保存
                swap.w  d4                  *
                move.w  ST+X(a6),d2         *(d2.w,d3.w)=始点座標
                move.w  ST+Y(a6),d3         *
                move.w  ST+E(a6),d4         *d4.w=始点移動経路決定用e

loop:
*               d2.l    始点,終点のx座標
*               d3.l    始点,終点のy座標
*               d4.l    始点,終点移動経路決定用e
*                       （上位ワード：終点/下位ワード：始点）
*               d5.w    始点（または終点）移動速度調節用e
*               d6.w    パターンを垂直になぞる速度の調節用e
*               d7.w    ループカウンタ
*               a1      参照中のパターン上水平線分左端アドレス
*               a3      始点を移動するサブルーチン
*               a4      終点を移動するサブルーチン
*               a5      パターン1ラインバイト数

                move.w  d2,d0               *d0=始点のx座標
                move.w  d3,d1               *d1=始点のy座標

                swap.w  d2                  *d2.w=終点のx座標
                swap.w  d3                  *d3.w=終点のy座標
                swap.w  d4

                bsr     put1line            *1ライン分描画する

                lea.l   ED(a6),a0           *終点を更新する
                jsr     (a4)                *

                swap.w  d2
                swap.w  d3
                swap.w  d4

                lea.l   ST(a6),a0           *始点を更新する
                jsr     (a3)                *

                add.w   PVDY(a6),d6         *移動速度を考慮して
                ble     nextln              *  パターンの垂直方向参照位置を
                                            *  移動する
skppat:         adda.l  a5,a1               *
                sub.w   PVDX(a6),d6         *
                bgt     skppat              *

nextln:         dbra    d7,loop             *必要なだけ繰り返す

                movem.l (sp)+,d0-d7/a0-a5
                unlk    a6
                rts

*
*               1ライン分描画する
*
put1line:
                movem.l d2-d7/a1/a3-a5,-(sp)

                jsr     gramadr             *a0=始点のG-RAMアドレス

                sub.w   d0,d2               *始点と終点を結ぶ線分を
                move.w  d2,d4               *  発生するための
                ABS     d2                  *  パラメータの初期化
                move.w  d2,a2               *
                add.w   a2,a2               *
                SGN     d4                  *
                                            *
                sub.w   d1,d3               *
                move.w  d3,d5               *
                ABS     d3                  *
                move.w  d3,a3               *
                add.w   a3,a3               *
                SGN     d5                  *

                move.w  d5,d6               *d6=y座標を更新したときの
                moveq.l #GSFTCTR,d7         *  G-RAMアドレスの変化量
                asl.w   d7,d6               *

                move.w  PHDY(a6),a4         *パターンを水平になぞる速度の
                                            *  調節用e増分

                cmp.w   d3,d2               *線分の傾きに応じて
                bcs     yput                *  処理を振り分ける

xput:                                   *なだらかな傾きの線分へ張り付ける場合
                move.w  d2,d7               *d7=ループカウンタ
                neg.w   d2                  *d2=誤差項e

                move.w  PHE(a6),d3          *パターンを水平になぞる速度の
                                            *  調節用e

xloop:          PSET    d0,d1,(a1),(a0)     *1ピクセル描画

                add.w   a4,d3               *移動速度を考慮して
                ble     xput1               *  パターンの水平方向参照位置を
                                            *  更新する
xput0:          addq.l  #2,a1               *
                sub.w   a2,d3               *
                bgt     xput0               *

xput1:          add.w   d4,d0               *x+=sx
                adda.w  d4,a0               *
                adda.w  d4,a0               *

                add.w   a3,d2               *e+=dy
                bmi     xnext

                add.w   d5,d1               *y+=sy
                adda.w  d6,a0               *

                sub.w   a2,d2               *e-=dx

xnext:          dbra    d7,xloop

                movem.l (sp)+,d2-d7/a1/a3-a5
                rts

yput:                                   *急な傾きの線分へ張り付ける場合
                move.w  d3,d2
                move.w  d2,d7
                neg.w   d2

                move.w  PHE(a6),d3

yloop:          PSET    d0,d1,(a1),(a0)

                add.w   a4,d3
                ble     yput1

yput0:          sub.w   a3,d3
                addq.l  #2,a1
                bgt     yput0

yput1:          add.w   d5,d1
                adda.w  d6,a0

                add.w   a2,d2
                bmi     ynext

                add.w   d4,d0
                adda.w  d4,a0
                adda.w  d4,a0

                sub.w   a3,d2

ynext:          dbra    d7,yloop

                movem.l (sp)+,d2-d7/a1/a3-a5
                rts

*
*               AB,DC発生のための初期化
*
init:
                sub.w   d0,d2               *d2=d4=x1-x0
                move.w  d2,d4               *
                ABS     d2                  *d2=dx=abs(x1-x0)
                SGN     d4                  *d4=sx=sgn(x1-x0)

                sub.w   d1,d3               *d3=d5=y1-y0
                move.w  d3,d5               *
                ABS     d3                  *d3=dy=abs(y1-y0)
```


▶武蔵野線が止まった。ただでさえ遠いうちの大学は埼玉県の奥地にある。片道3時間，往復6時間。1日の4分の1が電車の中。えっ，半年は無理？ あああそう，はあ……。

柴田　知久(19)神奈川県

```
        SGN     d5                      *d5=sy=sgn(y1-y0)

        cmp.w   d3,d2                   *傾きに応じて
        bcs     yinit                   *  処理を振り分ける

                        *dx≧dyの場合
xinit:  move.w  d2,d6                   *d6=dx
        neg.w   d6                      *d6=e=-dx
        add.w   d3,d2                   *d2=dx+dy
        move.w  d2,d7                   *d7=dx+dy
                                        *  =線分のピクセル数-1
        add.w   d2,d2                   *d2=2dx+2dy
                                        *  =y座標更新時のe補正値
        add.w   d3,d3                   *d3=2dy
                                        *  =x座標更新時のe増分
        movem.w d0-d7,X(a0)             *ワークに格納する

        move.w  d2,SDY(a0)              *速度調節用e増分
        lea.l   xlinex(pc),a1           *a1=傾きがなだらかな線分を
                                        *  1ピクセルずつ発生する
                                        *  サブルーチン

        rts

                        *dx<dyの場合
yinit:  move.w  d3,d6                   *d6=dy
        neg.w   d6                      *d6=e=-dy
        add.w   d2,d3                   *d3=dx+dy
        move.w  d3,d7                   *d7=dx+dy
                                        *  =線分のピクセル数-1
        add.w   d2,d2                   *d2=2dx
                                        *  =x座標更新時のe増分
        add.w   d3,d3                   *d3=2dx+2dy
                                        *  =y座標更新時のe補正値
        movem.w d0-d7,X(a0)             *ワークに格納する

        move.w  d3,SDY(a0)              *速度調節用e増分
        lea.l   ylinex(pc),a1           *a1=傾きが急な線分を
                                        *  1ピクセルずつ発生する
                                        *  サブルーチン

        rts

*
*       移動速度を考慮して始点,終点を移動する
*       (なだらかな傾きの線分用)
*
xlinex: add.w   SDY(a0),d5              *速度の調節
        bmi.s   xline1                  *

        sub.w   SDX(a0),d5              *

xline:  add.w   DY(a0),d4               *e+=2dy
        bmi     xline0

        add.w   SY(a0),d3               *y+=sy
        sub.w   DX(a0),d4               *e-=2(dx+dy)
        rts

xline0: add.w   SX(a0),d2               *x+=sx
xline1: rts

*
*       移動速度を考慮して始点,終点を移動する
*       (急な傾きの線分用)
*
ylinex: add.w   SDY(a0),d5              *速度の調節
        bmi.s   yline1                  *

        sub.w   SDX(a0),d5              *

yline:  add.w   DX(a0),d4               *e+=2dx
        bmi     yline0

        add.w   SX(a0),d2               *x+=sx
        sub.w   DY(a0),d4               *e-=2(dx+dy)
        rts

yline0: add.w   SY(a0),d3               *y+=sy
yline1: rts

        .end
```

リスト4 tformtest.s

```
*       gtputの動作試験用

        .include        doscall.mac
        .include        iocscall.mac
*
        .xref   gtput
*       .xref   setcliprect
*
FPACK   macro   callname
        dc.w    callname
        endm
*
__RAND  equ     $fe0e
*
        .text
        .even
*
ent:
        lea.l   inisp,sp

        lea.l   arg0(pc),a1
        IOCS    _GETGRM

        move.l  #$0010_0005,-(sp)
        DOS     _CONCTRL

        suba.l  a1,a1
        IOCS    _B_SUPER

*       pea.l   window(pc)
*       bsr     setcliprect
*       addq.l  #4,sp

        lea.l   arg(pc),a1
loop:   pea.l   (a1)
        bsr     gtput
        addq.l  #4,sp

        movea.l a1,a0
        moveq.l #8-1,d1
loop2:  FPACK   __RAND
*       lsr.w   #6,d0   *0≦d0≦511
        lsr.w   #5,d0   *0≦d0≦1023
        subi.w  #256,d0 *-256≦d0≦767
        move.w  d0,(a0)+
        dbra    d1,loop2

        DOS     _KEYSNS
        tst.l   d0
        beq     loop

        move.w  #-1,-(sp)
        DOS     _KFLUSH
        DOS     _EXIT
*
HSIZE   equ     128
VSIZE   equ     128
*
arg:    .dc.w   50,0,0,480,511,350,400,50
        .dc.l   pat
        .dc.w   HSIZE-1,VSIZE-1
*
arg0:   .dc.w   0,0,HSIZE-1,VSIZE-1
        .dc.l   pat
        .dc.l   pate
*
window: .dc.w   64,64,511-64,511-64
*
        .bss
        .even
*
pat:    .ds.w   HSIZE*VSIZE
pate:
*
        .stack
        .even
*
        .ds.l   2048
inisp:

        .end    ent
```

リスト5 gradpat.s

```
*       グラデーションパターンを作成する
*
        .include        gmacro.h
*
        .xdef   gradpat1
        .xdef   gradpat2
*
        .offset 0
*
BBUFF:  .ds.w   128
RBUFF:  .ds.w   128
GBUFF:  .ds.w   128
RGBBUF:
*
        .offset -RGBBUF*3
*
WORKTOP2:
T2:     .ds.b   RGBBUF
T1:     .ds.b   RGBBUF
WORKTOP1:
T0:     .ds.b   RGBBUF
_A6:    .ds.l   1
_PC:    .ds.l   1
BUFF:   .ds.l   1
COL1:   .ds.w   1
COL2:   .ds.w   1
COL3:   .ds.w   1
COL4:   .ds.w   1
*
        .text
```

▶中森章さんへ。酒井法子のスリーサイズはあるていど変化しています。しかし，バストだけ全然変わらないというのは……怪しい。　　横山　典俊(21)東京都

```
        .even
*
gradpat1:
        link    a6,#WORKTOP1
        movem.l d0-d7/a0-a3,-(sp)

        movem.w COL1(a6),d0-d1  *2色を補間して
        lea.l   T0(a6),a0       *  RGB別に
        bsr     calcgrad        *  テーブルを作成する

        movea.l BUFF(a6),a0     *RGBからカラーコードに
        bsr     genpat          *  再構成する

        movem.l (sp)+,d0-d7/a0-a3
        unlk    a6
        rts
*
gradpat2:
        link    a6,#WORKTOP2
        movem.l d0-d7/a0-a4,-(sp)

        movem.w COL1(a6),d0-d1  *COL1,COL2を補間して
        lea.l   T1(a6),a0       *  T1以降に
        bsr     calcgrad        *  中間パターンを作成する

        move.w  COL4(a6),d0     *COL4,COL3を補間して
        move.w  COL3(a6),d1     *  T2以降に
        lea.l   T2(a6),a0       *  中間パターンを作成する
        bsr     calcgrad        *

        movea.l BUFF(a6),a2     *a2=最終的なパターン
        lea.l   T1(a6),a3       *a3=中間パターン1
        lea.l   T2(a6),a4       *a4=中間パターン2

        moveq.l #128-1,d7
loop:   move.w  GBUFF(a3),d0    *緑成分を補間
        move.w  GBUFF(a4),d1    *
        lea.l   T0+GBUFF(a6),a0 *
        bsr     sub0            *

        move.w  RBUFF(a3),d0    *赤成分を補間
        move.w  RBUFF(a4),d1    *
        lea.l   T0+RBUFF(a6),a0 *
        bsr     sub0            *

        move.w  (a3)+,d0        *青成分を補間
        move.w  (a4)+,d1        *
        lea.l   T0+BBUFF(a6),a0 *
        bsr     sub0            *

        movea.l a2,a0           *RGBからカラーコードに
        bsr     genpat          *  再構成する
        movea.l a0,a2           *

        dbra    d7,loop

        movem.l (sp)+,d0-d7/a0-a4
        unlk    a6
        rts
*
calcgrad:
        DERGB   d0,d2,d3,d4     *COL1をRGBに分解
        DERGB   d1,d5,d6,d7     *COL2をRGBに分解

        move.w  d2,d0           *青成分を補間
        move.w  d5,d1           *
        bsr     sub             *

        move.w  d3,d0           *赤成分を補間
        move.w  d6,d1           *
        bsr     sub             *

        move.w  d4,d0           *緑成分を補間
        move.w  d7,d1           *
*       bsr     sub             *
*       rts

*
*       d0,d1を128段階で補間する
*
sub:
        lsl.w   #8,d0
        lsl.w   #8,d1

sub0:   movem.l d2-d5,-(sp)

        sub.w   d0,d1
        move.w  d1,d2
        ABS     d1
        SGN     d2

        add.w   d1,d1
        moveq.l #-127,d3
        move.w  #127*2,d4

        moveq.l #128-1,d5
sublp:  move.w  d0,(a0)+
        add.w   d1,d3
        bmi     next
sublp2: add.w   d2,d0
        sub.w   d4,d3
        bpl     sublp2
next:   dbra    d5,sublp

        movem.l (sp)+,d2-d5
        rts

*
*       RGB別のテーブルから
*       カラーコードに再構成する
*
genpat:
        lea.l   T0(a6),a1

        moveq.l #128-1,d4
genlp:  move.w  GBUFF(a1),d3
        move.w  RBUFF(a1),d2
        move.w  (a1)+,d1

        lsr.w   #8,d1
        lsr.w   #8,d2
        lsr.w   #8,d3

        RGB     d1,d2,d3,d0
        move.w  d0,(a0)+

        dbra    d4,genlp
        rts

        .end
```

リスト6 gradtest.s

```
*       gtputの動作試験用

        .include        doscall.mac
        .include        iocscall.mac
*
        .xref   gtput
*       .xref   setcliprect
*
FPACK   macro   callname
        dc.w    callname
        endm
*
__RAND  equ     $fe0e
*
        .text
        .even
*
ent:
        lea.l   inisp,sp

        lea.l   arg0(pc),a1
        IOCS    _GETGRM

        move.l  #$0010_0005,-(sp)
        DOS     _CONCTRL

        suba.l  a1,a1
        IOCS    _B_SUPER

*       pea.l   window(pc)
*       bsr     setcliprect
*       addq.l  #4,sp

        lea.l   arg(pc),a1
loop:   pea.l   (a1)
        bsr     gtput
        addq.l  #4,sp

        movea.l a1,a0
        moveq.l #8-1,d1
loop2:  FPACK   __RAND
*       lsr.w   #6,d0   *0≦d0≦511
        lsr.w   #5,d0   *0≦d0≦1023
        subi.w  #256,d0 *-256≦d0≦767
        move.w  d0,(a0)+
        dbra    d1,loop2

        DOS     _KEYSNS
        tst.l   d0
        beq     loop

        move.w  #-1,-(sp)
        DOS     _KFLUSH
        DOS     _EXIT
*
HSIZE   equ     128
VSIZE   equ     128
*
arg:    .dc.w   50,0,0,480,511,350,400,50
        .dc.l   pat
        .dc.w   HSIZE-1,VSIZE-1
*
arg0:   .dc.w   0,0,HSIZE-1,VSIZE-1
        .dc.l   pat
        .dc.l   pate
*
window: .dc.w   64,64,511-64,511-64
*
        .bss
        .even
*
pat:    .ds.w   HSIZE*VSIZE
pate:
*
        .stack
        .even
*
        .ds.l   2048
inisp:

        .end    ent
```

ファイル入出力って何だろう

〔最終回〕

Nakamori Akira
中森 章

1年以上にわたって続けてきたこの連載も，一応今回でひと区切り。中森氏の懇切丁寧な解説で，皆さんもC言語というものを理解することができたことと思います。これからも，皆さんがC言語に興味を持って接してくれることを祈っています。

最近は昔のアニメの全話LD（やビデオ）の発売が盛んです。この手のLDやビデオは「イデオン」と「めぞん一刻」しか持っていないのですが，お店で「クリーミィマミ」の全話LDを見るたびに衝動買いしたい欲求を抑えるのにひと苦労する中森章です。これが「ミンキーモモ」なら何も考えないで買ってしまうのですが……。

さて，今回はファイルの入出力を取り上げます。ファイルはコンピュータの世界では常識です。ファイルはデータを保存する手段です。ファイルは，メモ用紙であり，プログラムであり，コマンドであり，私たちがコンピュータの資源で直接アクセスできるもののすべてなのです。当然，どのようなプログラミング言語もファイルを扱う機能を有しています。ファイルとの入出力がアプリケーションによっては不可欠な機能になることもあります。

しかし，C言語はファイルの入出力を規定しないプログラミング言語なのです。いや，入出力を規定していなかったというほうが正確でしょう。入出力はプログラムが動く処理系の実装方式と深く関わっています。このため，C言語では実装方式の異なる入出力の部分を文法として規定せず，各処理系が独自のライブラリ関数を作って対応することを期待していた感があります。私事をいえば，UNIXのC言語とパソコンのBDS-CやSmall-Cで入出力のやり方が異なっていて，非常にとまどった経験があります。このような状況では他機種への移植性など生まれてくるはずはありません。

しかし，ANSI Cでは入出力のライブラリ関数の仕様が厳密に規定されています。これは，事実上の標準になっていたUNIXでの，高水準な入出力関数を元にした仕様です。したがって，UNIXのユーザーは何も考えなくても簡単にANSI Cに移行することができるのです。また，これらのライブラリ関数を使用している限り，C言語のプログラムが任意の処理系の上で互換性を持つことになります。

というわけで，C言語の学習において，以前はそれほど重視されなかった入出力関数も，最近ではぜひとも覚えるべき項目のひとつになっています。入出力関数はC言語の最後の砦です。がんばって学習していくことにしましょう。

ファイルとは何か

プログラムの構造を思い出しましょう。プログラムとは入力を処理して結果を出力するものでした。処理とは変数に格納された値を加工することです。C言語では既存のデータ型や構造体を組み合わせることでいろいろなデータ型を作り出すことができます。このデータ型の多彩さが複雑な変数の処理を可能にし，自由度の高いプログラムを作り出しています。しかし，そこには問題があります。

C言語（に限らず，すべて）のプログラムやデータはコンピュータのメモリにロードして実行されています。変数の領域はメモリに割り付けられ，その領域の内容を変更しながらプログラムの実行が進んでいきます。ただし，メモリはプログラムの実行が終了するとOSに開放されてしまいますから，プログラムの実行後ではせっかく計算した変数の値を参照することができなくなってしまいます。もし，プログラムの実行結果（変数の値）をあとから参照する必要があるならば，それをなんらかの形でフロッピーディスクやハードディスクなどの外部記憶装置に保存しておかなければなりません。また，出力結果を保存しておく以上に入力データを保存しておくことが重要です。

私たちがプログラムを書いてコンピュータに処理をさせる理由のひとつは，大量のデータを効率的に扱うためです。実際，請求書などの伝票処理を行うときに必要な情報をキーボードから入力し直すのは現実的ではありません。できることなら，外部記憶装置に保存してあるデータを読み込んで一気に処理をしたいものです。このように，プログラムの種類によっては変数の値を保存したり値を取り出すための仕組みが必要になります。それを実現するために，外部記憶装置に名前を付けて記憶したデータの集まりがファイルと呼ばれるものなのです。

ここではC言語のプログラムでの必要性から説明してきましたが，ファイルの重要性はそれだけにとどまりません。ファイルはコンピュータがものごとを処理するための基本的な単位なのです。あるいは，コンピュータが

管理するデータの集まりすべてがファイルということができます。たとえば，エディタのプログラムもファイルですし，エディタを使って作った（ソース）プログラムもファイルです。Cコンパイラのプログラムもファイルですし，ソースプログラムをコンパイルしてできる実行形式のプログラムもファイルなのです。また，X68000のOSであるHuman68kではキーボード，プリンタ，OPM，PCMなどの周辺機器もファイルとみなして扱うようになっているのです。このように，ファイルとは非常に広い意味を持った概念といえます。ファイルはコンピュータというものと切り離して考えることはできないのです。そのため，C言語がファイルを扱う機能を備えているのはプログラミング言語として当然のことなのです[1]。

1）ファイルの操作に限らず周辺装置との入出力の機能はC言語の文法では規定がない。入出力はライブラリ関数で実現することになっている。これは，K&Rの第1版の時代では処理系依存ということで，互換性のもっとも低い部分だった。ANSI Cでは，入出力を行うライブラリ関数に関して仕様の統一がなされており，それを使っているプログラムは異なる処理系の間での互換性を保証している。

入出力ストリーム

多くのプログラミング言語と同じく，C言語にもファイルを扱うための機能が用意されています。ファイルの基本的な操作は，

(1)　ファイルのオープン（扱うファイルの指定）
(2)　ファイルの読み書き（扱うファイルの操作）
(3)　ファイルのクローズ（扱うファイルの開放）

の3つです。この3つの操作が標準的なライブラリ関数として提供されているのです。

ところで，ひとつのプログラムで同時に扱うファイルの数はひとつとは限りません。このとき，プログラムでどのファイルに対して処理を行うのかを示すための情報が必要です。このため，C言語では扱うファイルを指定するための情報としてストリーム（流れ，という意味）というものを利用します。ストリームとは，プログラムとファイルの間のデータの流れを比喩的に表現したものです。ファイルの実体はハードディスクやフロッピーディスクなどの外部記憶装置にあるのですが，C言語のプログラムでファイルの内容を参照する（要するに読み書きする）ときは，ファイルと1対1に対応するストリームを媒介として参照する約束になっています。ファイルをオープンすることは，プログラムの側で新たなストリームを作ることにほかなりません。

なお，C言語ではわざわざオープンしなくても使用できるストリームが3種類存在します[2]。これらは，非常によく使われるストリームなので，プログラムの実行開始時に自動的にオープンされるのです。その中の2つが，第10回「標準入出力って何だろう」でも説明した，標準入力（stdin）と標準出力（stdout）です。残りが標準エラー出力（stderr）です。標準入出力についての説明は重複になるのでここでは省略します。標準エラー出力は，名前のとおりエラーメッセージを出力するためのストリームです。これは通常端末に割り当てられています[3]。

標準エラー出力が標準出力とわざわざ分けて用意されている理由は，標準出力を何かのファイルにリダイレクトしているときでも，エラーメッセージを見られるようにするためです。もし，エラーメッセージが標準出力に書かれるのなら，プログラムの処理結果と一緒にエラーメッセージまでもリダイレクトされているファイルに書き込まれてしまって，エラーが発生したこと（あるいは，どのような種類のエラーが発生したのか）を見落としてしまうかもしれません。

さて，ストリームというものは抽象的な概念ですが，もっと具体的なイメージとしてとらえることも可能です。プログラムというものは概念という実体のないものを操作するのではなく，メモリ上に領域が確保されている変数や配列といった実体を操作します。ストリームにも実体が存在するのです。それは，stdio.hというヘッダファイルの中で定義されているFILEという構造体です。具体的には，

```
typedef         struct    _iobuf    {
        char    *_ptr;
        int     _cnt;
        char    *_base;
        int     _flag;
        int     _bsize;
        char    _file;
        char    _pback;
        char    *_fname;
    }   FILE;
```

という定義の構造体です[4]。一見しただけではなんのことやらわかりませんが，ファイルを管理するためにはこれだけの情報が必要になるということです（構造体の名前_iobufから察すると，ファイルのバッファ制御をするための情報みたいですね）。この構造体の各メンバの意味を理解する必要はありませんが，FILEという構造体がストリームを制御しているのだということだけは覚えておきましょう。

C言語のプログラムでは，すべてのファイルはこのFILEという構造体へのポインタとして管理されます。つまり，ファイルをオープンするとFILE構造体へのポインタが作られます。以後，そのファイルを参照するためにはFILE構造体へのポインタを指定して読み書きを行うことになります。そして，最後にファイルをクローズすることで，FILE構造体へのポインタを無効化します。

ところで，ANSI規格では，ファイルのストリームに関し，テキストストリームとバイナリストリームの2種類を用意しています。これは，OSの環境によってはテキス

トファイルとバイナリファイルでファイル形式が異なることがあるからです。C言語発祥の地であるUNIXでは，テキストファイル，バイナリファイルでファイル形式の区別は特になかったのですが，そのほかのOSであるHuman68kやMS-DOSなどでは，テキストファイルとバイナリファイルで形式が少し異なっています。おそらく，ANSI規格ではそこら辺の事情を考慮してあるのでしょう。テキストストリームは行単位でファイルを扱うストリームです。これはテキストファイルを読み書きするためのストリームです。ファイルの形式の詳細は処理系依存ということになっています[5)]。一方，バイナリストリームはバイト単位でファイルを扱うストリームです。これは，ファイルを単純に8ビットのコードが並んだものと見なすだけなので，処理系による違いはありません。

ここで，Human68kやMS-DOSにおけるテキストストリーム（テキストファイル）とバイナリストリーム（バイナリファイル）の具体的な違いについて説明しておきましょう。C言語でファイルを扱う場合，それがテキストファイルであるかバイナリファイルであるかは重要です。場合によってはテキストファイルをバイナリファイルとみなして扱う場合も出てきますから，両者の違いをしっかりと理解しておかなければなりません。昔からいわれるテキストファイルとバイナリファイルの違いは，TYPEコマンドで画面に内容を正しく表示させることのできるのがテキストファイルで，画面がめちゃくちゃになるのがバイナリファイルということになりますが，この説明では何が本質的なのかはっきりしません。

Human68kやMS-DOSのテキストファイルは，基本的には英数字や漢字などの文字コードと改行コード，エンド・オブ・ファイル（ファイルの終了）コードからなるファイルです。テキストファイルではこの改行コード，エンド・オブ・ファイルコードがあることが特徴です。Human68kやMS-DOSのテキストファイルでは0x0D，0x0A（C言語のエスケープシーケンスで書けば¥r¥n）という連続2バイトが改行コードになっています（正確には0x0Dが復帰，0x0Aが改行だが）。UNIXでは0x0A（¥n）1バイトだけで改行コードです。Human68k，MS-DOSとUNIXでテキストファイルを扱うプログラムを変更することなく使うためか，テキストストリームでは改行コードの変換が行われます。つまり，0x0D，0x0Aという2バイトをテキストストリームから読み出すと0x0Aという1バイトになります。逆に，0x0Aという1バイトをテキストストリームに書き込むと0x0D，0x0Aという2バイトになるのです。バイナリストリームではこのような変換は行われず，ストリームに読み書きするコードは1バイトずつ1対1に対応するようになっています。これがテキストストリームとバイナリストリームの最大の違いです。このほか，テキストストリームとバイナリストリームではエンド・オブ・ファイルの扱いが異なります。

ファイルというものはバイトコードの集まりですから，大きさというものが存在します。ファイルの内容を先頭から1バイトずつ読んでいけば，ファイルの大きさを超えて内容を読むことはできませんから，最後にデータが尽きてしまいます。これがファイルの終わり，つまりエンド・オブ・ファイルです。これはファイルの物理的な終わりということになります。しかし，ファイルには物理的な終わりのほかに仮想的な終わりというものがあります。バイナリファイルでは仮想的な終わりは物理的な終わりと一致していますが，テキストファイルでは必ずしも一致しません。

テキストファイルにはエンド・オブ・ファイルという特殊なコードがあります。これは0x1A（CTRL-Z）というコードです。このコードをテキストストリームから読み出すと，そのテキストストリームはデータが尽きたと認識されるのです（テキストファイルでは0x1Aコードの後ろは存在していても読めないのです）。このエンド・オブ・ファイルのコードは，ファイルの大きさをクラスタ単位でしか管理していなかったCP/M時代の名残（という説が有力です）で，ファイルの大きさをバイト単位に管理するようになったHuman68kやMS-DOSでは特に意味のないものになっています。実際，テキストファイルの最後にエンド・オブ・ファイルのコードがなくても，ファイルの終わりは正しく認識されます。第10回「標準入出力って何だろう」で，標準入出力がテキストストリームとなっていることを示唆してありますが，標準入出力のリダイレクトでもエンド・オブ・ファイルのコードは邪魔になります。エンド・オブ・ファイルのコードで標準入力が尽きるのでなければ，バイナリファイルのリダイレクトも可能だったかもしれません。本当にエンド・オブ・ファイルのコードはなんのために存在するのでしょうね。と，疑問を残しつつ次に進みましょう。

---

2)　Human68kではstdin, stdout, stderrのほかにstdaux, stdprnというストリームがオープンされる。stdauxとは標準補助入出力を意味する。早い話がRS-232C。stdprnは標準プリンタ出力を意味する。これはプリンタである。それぞれ，AUXおよびPRNというファイルをオープンしてストリームを作っても同じことである。

3)　Human68kではstderrをファイルにリダイレクトすることは簡単ではない。UNIXでは簡単に標準出力と標準エラー出力を分離して別々のファイルにリダイレクトすることができる。

4)　これはXCのVer.2.0のもの。XCのVer.1.0では最後のfnameというメンバがない。FILE構造体に互換性がないので，XCのVer.1.0とVer.2.0でstdio.hとFILE構造体を扱う関数が含まれたライブラリを一致させないと，バスエラーなどの障害が発生する。また，FILE構造体は入出力の処理系依存の部分を吸収しているので，FILE構造体のメンバを直接操作するプログラムは，移植性の面で好ましくない。

5)　ANSI規格で入出力関数の仕様を決定しても，結局は処理系依存の非互換部分が残ってしまう。しかし，テキストストリームなどという概念（UNIXにはない）が導入されたのは，おそらくIBM PCの普及で圧倒的な占有率を誇るMS-DOSのテキストファイル形式を意識してのことと考えられる。MS-DOSのテキストファイルの形式が事実上の標準みたいなものなので互換性の問題は生じないのかもしれない。とはいえ，UNIXもMS-DOSも基本はASCIIコードなので大差ないが，IBMの大型機などのようにEBCDICコードを採用している処理系では問題かな。

ファイルを扱うための関数

それでは，C言語でファイルの入出力を行うためのライブラリ関数について説明します。先に示したように，ファイルの操作というものは，オープン，読み書き，クローズという手順で行われます。ファイルをオープンすると，そのファイルに対してFILE構造体へのポインタが作られ，そのポインタを指定することで読み書き，クローズの操作をすることになります。これを頭に置いたところで，それぞれの段階で使用される関数を以下に説明していきます。

なお，これらの関数を使うためにはFILEという構造体の定義や，関数の返り値の宣言が必要なので，それらを記述してあるstdio.hというヘッダファイルをプログラムに取り込まなければまりません。そのためには，プログラムの先頭に，

```
#include <stdio.h>
```

という1行を付け加えておく必要があります。

(1)ファイルのオープン

fopen関数は，オープンするファイル名とオープンのモード（仕方）を指定します。そして，指定したファイルをストリームに割り当てて，FILE構造体へのポインタを返します。もし，なんらかの理由でファイルがオープンできないときはNULL（値0）を値として返します。以下にfopen関数の形式を示します。

・FILE ＊fopen(file,mode)

char ＊file　オープンするファイル名
OSで許されている名前を指定する

char ＊mode　オープンするモード
“r”“w”“a”“t”“b”“＋”の文字の組み合わせで指定する。“t”“b”以外は次の組み合わせが有効

- ”r”　リードのみ，ファイルは存在すること
- ”w”　ライトのみ，ファイルを新規作成。同じ名前のファイルは書き潰し
- ”a”　ファイルの最後に追加，ライトのみ。ファイルが存在しないと新規作成
- ”r＋”　リード/ライト，ファイルは存在すること
- ”w＋”　リード/ライト，ファイルを新規作成。同じ名前のファイルは書き潰し
- ”a＋”　ファイルの最後に追加，リード/ライト。ファイルが存在しないと新規作成

“t”“b”は次の意味を持つ

- ”t”　ファイルをテキストファイルとして扱う
- ”b”　ファイルをバイナリファイルとして扱う

ファイルのモードのうち，“t”または“b”は，“r”“w”“a”“r＋”“w＋”“a＋”の任意の位置に記述することができます。たとえば，

”rb”
”br”　ANSI仕様では不可
”w＋b”
”wb＋”

といったものが指定可能です。“t”と“b”が同時に指定された場合は，文字列の中で先に指定したものが有効になるようです[6]。また，“t”も“b”も指定されないときは，システムの規定値が使用されます。Human68kでは規定値はテキストファイルになっているようです。

(2)ファイルのリード/ライト

ファイルの入出力用の関数は「文字」，「直接」，「書式付き」の3種類に分類されます。「文字」は，入出力ストリームに対し，1文字単位，あるいは1行単位（テキストファイルの場合）で入出力を行うための関数です。「直接」は，入出力ストリームに対してあるまとまったサイズを単位として入出力を行う関数です。これは，構造体や配列などまとまったサイズを持つ変数とデータのやり取りを行うための関数です。「書式付き」は，ある文字列で指定される書式に従って，入出力ストリームのデータに変換を加えながら入出力を行うための関数です。おなじみのprintf関数，scanf関数はこの「書式付き」の入出力関数に分類されます。

以下に入出力を行う関数のうち，主なものを示します。非常にたくさんの関数があるように見えますが，基本は第10回「標準入出力って何だろう」で説明したscanf, gets, getchar, printf, puts, putcharという関数です。これらは，すべて標準入出力ストリームに対して読み書きを行うものでしたが，これらがFILE構造体へのポインタで指定されるストリームに変わっただけのものです。そのため，関数の引数としてFILE構造体へのポインタが付け加わっているのがわかると思います。そして，それを示すために関数の名前の先頭にfの文字（ファイルに対する，という意味なのでしょう）が付いているだけなのです(getc，putcは例外)。fread，fwrite以外の関数は第10回「標準入出力って何だろう」で説明した関数とストリームが異なるだけなので，ここでは簡単な説明にとどめます[7]。詳しくは第10回目の連載を見直してください。fread，fwriteについては，あとで示すプログラム例で具体的な使い方を見てください。

■ **文字**

・int getc(fp)

FILE　*fp　FILE構造体へのポインタ
機能　ストリームから１文字読み込む。戻り値は読んだ文字

・int getchar( )
機能　標準入力から1文字読み込む。戻り値は読んだ文字

・int putc(ch,fp)
int ch　文字
FILE *fp　FILE構造体へのポインタ
機能　ストリームに１文字書き込む。戻り値は書いた文字であるが，ふつう参照しない

・int putchar(ch)
int ch　文字
機能　標準出力に１文字書き込む。戻り値は書いた文字であるが，ふつう参照しない

・char *fgets(str,n,fp)
char *str　文字列を格納する配列名，あるいはポインタ
int n　読み込む１行のバイト数
FILE *fp　FILE構造体へのポインタ
機能　ストリームから１行読み込む。戻り値は読んだ文字列を格納する配列へのポインタ

・char *gets(str)
char *str　文字列を格納する配列名，あるいはポインタ
機能　標準入力から１行読み込む。戻り値は読んだ文字列を格納する配列へのポインタ

・int fputs(str,fp)
char *str　文字列が格納された配列名，あるいはポインタ
FILE *fp　FILE構造体へのポインタ
機能　ストリームに１行書き込む。エラーのときEOF，そうでなければ非負の値

・int puts(str)
char *str　文字列が格納された配列名，あるいはポインタ
機能　標準出力に１行書き込む。エラーのときEOF，そうでなければ非負の値

■**直接**

・int fread(str,size,n,fp)
char *str　データを読み込む配列名，あるいは構造体へのポインタ
int size　読み込むデータの単位長
int n　読み込むデータの個数
FILE *fp　FILE構造体へのポインタ
機能　ストリームからまとまったデータを読み込む。戻り値は読んだデータの個数

・int fwrite(str,size,n,fp)
char *str　データを書き込む配列名，あるいは構造体へのポインタ
int size　書き込むデータの単位長
int n　書き込むデータの個数
FILE *fp　FILE構造体へのポインタ
機能　ストリームにまとまったデータを書き込む。戻り値は書いたデータの個数

■**書式付き**

・int scanf(fmt,…)
char *fmt　変換書式
機能　書式を指定して，テキストを標準入力から読み込む。戻り値は読んだデータの個数

・int fscanf(fp,fmt,…)
FILE *fp　FILE構造体へのポインタ
char *fmt　変換書式
機能　書式を指定して，テキストをストリームから読み込む。戻り値は読んだデータの個数

・int printf(fmt,…)
char *fmt　変換書式
機能　書式を指定して，テキストを標準出力に書き込む。戻り値は書いた文字数

・int fprintf(fp,fmt,…)
FILE *fp　FILE構造体へのポインタ
char *fmt　変換書式
機能　書式を指定して，テキストをストリームに書き込む。戻り値は書いた文字数

⑶ファイルのクローズ

fclose関数は，引数で与えられるFILE構造体へのポインタで示されるストリームをクローズします。ファイルが正しくクローズできるとfclose関数は０を返し，そうでなければEOF（－１）を返します。また，fcloseall関数はオープンしてある（stdin，stdout，stderr，stdaux，stdprn以外の）すべてのストリームをクローズします。正常にクローズできた場合はクローズしたストリームの数を返し，そうでなければEOFを返します。ただし，fcloseall関数はANSI Cでは規定されていないので，この関数を使用する場合は互換性に注意する必要があります。fclose関数，fcloseall関数の形式は次のようになっています。

・int fclose(fp)

　FILE ＊fp　ストリームを示すFILE構造体へのポインタ

・int fcloseall( )

---

6)　ANSI Cのfopenの仕様では"t"（テキストモード）の指定はない。"b"(バイナリモード)を指定しないとテキストモードとみなされる。
7)　getcとfgetc, putcとfputcは同じ機能を持つ。f の付かないgetcとputcは，処理系によっては(実行速度の向上のため)プリプロセッサのマクロ機能で実現される(UNIXではマクロで実現されている)。マクロで実現された場合，副作用を生じる式を引数に与えると誤動作することもある。f の付いたfgetcとfputcは，関数であることが保証されているので，マクロ展開されると困る場合はこちらを使用する。XCでこれらの関数をマクロで使う場合は，

　#include ＜stdio.h＞

の前の行に，

　#define MACRO

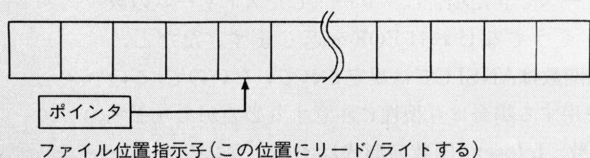
を指定しておけばよい。

シーケンシャルファイルとランダムファイル

C言語では,ファイルはFILE構造体へのポインタ(K&Rではファイルポインタと呼んでいる）によって管理されます。各ファイルのFILE構造体へのポインタはfopen関数の戻り値で与えられ,以後,そのファイルに対するリード/ライト,あるいはクローズはそれらの処理を行う関数にそのFILE構造体へのポインタを引数として与えることで行われることになります（ここまでは復習ね)。また,FILE構造体へのポインタは,ファイル内で読み書きが行われている位置の情報（ファイル位置指示子,XCのマニュアルではファイルポインタと呼んでいるが,K&Rでいうファイルポインタとの違いに注意)を含んでいます。それは,通常はファイルに対してリード/ライトが行われるたびに増加していきます。当然のことながらファイルがオープンされた直後では，このファイル位置指示子はファイルの先頭を指し示しています（追加モードでオープンしたときはファイルの最後を指しています)。C言語におけるファイルというものは，仮想的に,図1のような構造を持ったものと考えることができます。

通常は，ファイルの入出力はファイル位置指示子が指し示す位置に対して行われます。つまり，ファイルの処理は先頭から逐次的にリードまたはライトされていくことになります。このように，ファイルの先頭から逐次的に処理を行っていき，あと戻りして参照することのないようなファイル構造をシーケンシャルファイル（正確にはシーケンシャルアクセスファイル）と呼びます。ファイルの処理は，多くの場合，逐次的な処理だけでこと足ります。アセンブラやコンパイラでのソースファイル(ソースプログラム）の処理，Human68kのTYPEコマンドやCOPYコマンドによるファイルの表示や複製が逐次的な処理の代表例です。

図1　ファイルの仮想的な構造



しかし，アプリケーションの種類によってはファイルを逐次的に参照していたのでは実現できないような処理もあります。たとえば，テキストエディタで編集するファイルやデータベースが扱うデータファイルは，ファイル内の任意の位置をプログラムで参照する必要があります。このようにファイルの内部を自由に参照して処理を行うようなファイル構造をランダムファイル（正確にはランダムアクセスファイル）と呼びます。

C言語でランダムファイルを実現するためには，先に述べたファイル位置指示子を直接操作することによって行うことができます。これは，プログラムの中にカウンタなどを持っていて，1文字ずつ空読みをしながらファイル位置指示子を目的の場所に持っていくことではありません(そりゃそうだ)。直接ファイル位置指示子を目的の場所に設定して，その場所を参照するのです。C言語ではファイル位置指示子の値を知るためにはftell関数,ファイル位置指示子を設定するためにfseek関数が用意されています[8]。これらの関数（特にfseek関数）を使用することで，ランダムファイルを実現するのです。これらの関数の形式を以下に示しておきましょう。

・int ftell(fp)

　FILE ＊fp　ストリームを示すFILE構造体へのポインタ

　機能　ファイル位置指示子の値を返す

・int fseek(fp,offset,whence)

　FILE ＊fp　ストリームを示すFILE構造体へのポインタ

　int offset　ファイル位置指示子の基準位置(whence)からの移動量。正数なら基準位置より後方に，負数なら基準位置より前方に移動する

　int whence　移動の基準位置。0，1，2で指定する[9]

　　0…ファイルの先頭
　　1…ファイル位置指示子の現在位置
　　2…ファイルの終端

ところで，ファイル位置指示子の操作は，バイナリファイルとテキストファイルで動作が異なることがあります。一般に，バイナリファイルを扱うときには，ファイル位置指示子は1文字読み書きするたびに1ずつ増加していきます。しかし，テキストファイルでは改行文字の変換などの処理があるため，ファイル位置指示子の操作は処理系依存になっているのです。テキストファイルに関するfseek関数に関して，ANSI規格では，

・offsetが0である移動(つまり，ファイルの先頭や終端への移動)
・ファイル先頭からの，ftell関数で返されたoffsetまでの移動（つまり，以前ftell関数で記憶していた位置への移動)
の2つの操作での結果は保証しています。XCでもこれ以外の操作では動作が保証されていません。

8) ANSI Cではファイル位置指示子の値がlong型で表せない場合を想定して，ファイル位置指示子を操作する関数として，fseek関数のほかにfgetpos関数，fsetpos関数を用意している。XCではVer.2.0でサポートされた。これらの関数では，ファイル位置指示子はfpos_tというデータ型で表される（これは4Gバイト以上のサイズのファイルを扱える処理系用ですね)。XC(だけでなくたいていの処理系)ではlong型と同じである。
9) XCのVer.2.0では，stdio.hというヘッダファイルをインクルードすることにより，0，1，2の代わりにSEEK_SET, SEEK_CUR, SEEK_ENDという定数で基準位置を設定できる。この定数を使用するほうがANSI Cに準拠している。

ファイルの終了判定

ファイルの入力関数を使ってファイルのリードをやっているとき，ファイルの終端（エンド・オブ・ファイル）に到達するとNULL（値0）なりEOF（値－1）なりの値が関数の戻り値になるので，入力データが尽きたことが判断できます。第10回「標準入出力って何だろう」で示したプログラム例では，入力データが尽きるまで読み込むための処理として，

```
while((ch=getc( ))!=(−1)) { … }
```

とか，

```
while(gets(BUF)!=0) { … }
```

という記述がありますが，これらはエンド・オブ・ファイルで特殊な値が戻り値となることを利用した記述です。しかし，厳密な意味ではこういった記述では不十分です。もし，このようにプログラムを書いて，それをそのままプログラムコンテストなどに応募するとしたら減点対象です。その理由は，マニュアルを読めばわかりますが，ファイルの入力関数でNULLとかEOFといった値が返ってくるのは，

エンド・オブ・ファイルに達した場合

あるいは，

エラーが発生した場合

だからです。つまり，NULLまたはEOFでエンド・オブ・ファイルと判断するとエラーの発生を見落とすことになるからです。アプリケーションの種類によっては，エラーが発生した場合にファイルの読み直しをしなければなりません。

そこで，エンド・オブ・ファイルなのかエラーなのかを知るためにC言語ではfeof，ferrorという関数が用意されています[10]。これらは次のような形式で使用します。

・int feof(fp)

FILE *fp	ストリームを示すFILE構造体へのポインタ
機能	指定したストリームがエンド・オブ・ファイルでないなら0，エンド・オブ・ファイルなら0以外の値を返す

・int ferror(fp)

FILE *fp	ストリームを示すFILE構造体へのポインタ
機能	指定したストリームでエラーが発生していないのなら0，エラーが発生していれば0以外の値を返す

つまり，上の例を完全にするなら，

```
while((ch=getc( ))!=(−1)) { … }
if(ferror(stdin)) { エラー処理 }
```

というように書くのが正しい処理なのでしょう。標準入力のエンド・オブ・ファイルやエラーを調べるためには，stdin（これはシステムであらかじめ用意されるFILE構造体へのポインタ）をfeof関数やferror関数の引数に指定します。ただし，現実問題としてファイルのリードでエラーが発生することはほとんどありませんし[11]，ファイルがそれ以上読めないことには変わりありませんので，ちょっとした練習程度のプログラムならばこんな大袈裟な仕掛けは不要かもしれません。

10) XCVer.2.0のライブラリマニュアルでのferror関数の説明での使用例は誤り。

```
while((c=getc(fp))!=EOF) {
    if(ferror(fp)) { … }
}
```

などとなっているが，whileループの内部を実行するときはストリームがEOFになっているわけがない（エンド・オブ・ファイルにもエラーにもなっていない）ので，

```
if(ferror(fp))
```

などという判断自体が無意味である。
11) エラーが発生するとしたら，ディスクのセクタが壊れているか，CRCエラーの場合であろう。どちらの場合もディスクコントローラが何度かリトライ（再試行）を行ってもだめだった結果であると予想される。そんな状況で，ファイル位置指示子をファイルの先頭に巻き戻して(rewind関数を使う)再びファイルを読んでみるというエラー処理を行っても，同じ場所でエラーが発生するのは明白である(CRCエラーなら大丈夫かもしれないが)し，標準入力など巻き戻しの効かないストリームもあるので無駄なあがきはしなくてもよいと思う。これは，本当に個人的な意見。もちろん違う考えの人もいるだろう。

ファイルを扱うプログラム

前置きが長くなりましたが，それでは実際にファイルを参照するプログラムを書いてみましょう。ファイルにはシーケンシャルファイル，ランダムファイルという区別があることはすでに説明しました。まずシーケンシャルファイルを扱う例をいくつか示してから，最後にほんの少しだけランダムファイルの例を示すことにしましょう。

●1バイト単位の入出力

シーケンシャルファイルを扱うプログラムのもっとも典型的な例は（テキスト）ファイルの内容を画面にプリントするものでしょう（要するにHuman68kのtypeコマンドみたいなものですね）。小手調べにこのプログラムを作ります。プログラムの動作としてはファイルから読み込む1文字を，エンド・オブ・ファイルになるまで，そのまま順番に画面に表示（標準出力に書き込む）すればよいので，プログラムはリスト1のようになります。ファイルの入出力を扱うということで，

#include 〈stdio.h〉

という1行を入れてあります。リスト1のプログラムでは特にエラーチェックをしていませんが，そのためやっていることがより明白になっていると思います。つまり，main関数への引数，つまりコマンドラインで与えたコマンドへの引数で指定されるファイルをオープンしてその内容を標準出力に書き込んでいるのです。もし，リスト1のプログラムをコンパイルした結果がtypec.xという名前になっているのなら，コマンドラインから，

typec ファイル名

と入力すれば，“ファイル名”で指定されるファイルの内容が画面に表示されます。コマンドラインで渡す引数については第12回「ポインタって何だろう（後編）」で説明してありますから詳しくはそちらを見てください。

ところで，リスト1ではエラーチェックをまったくしていませんが，ファイルがオープンできたかどうかのチェック（fopen関数の戻り値のチェック）は行うほうが賢明です。なぜならfopen関数がエラーになると（指定したファイルが存在しない場合など）NULL（値0）が返ってきます。つまり，FILE構造体へのポインタfpの値は0になります。値が0であるFILE構造体へのポインタをファイル入出力関数の引数に指定するとバスエラーが発生するので，これは避けたほうがいいでしょう。

なお，プログラムで入力ファイルがオープンできない場合，入力ストリームを標準入力に切り替えてしまうということも可能です。これはfopen関数でNULLが返ってきた場合，FILE構造体へのポインタに標準入力を示すstdinの値を代入することで実現できます。つまり，ファイルをオープンする部分で，

fp＝fopen(argv［1］,”r”);

if(fp＝＝NULL) fp＝stdin;

というような記述をしておけばよいのです。こうすることで，入力ファイルがオープンできない場合は，入力データが標準入力から読み込まれるようになります。Human68kのコマンドであるmore.xなどは，入力ファイルをコマンドラインの引数で指定することも，標準入力から指定することも可能です。このような処理を実現するのがストリームを標準入力に切り替えることの目的なのです。つまり，このような動作をするプログラムがtypec.xという名前であるなら，

typec ファイル名

typec ＜ ファイル名

sort ファイル名 ｜ typec

といった使い方ができるようになります。いちばん上が通常の使い方，真ん中が標準入力からデータを与える使

リスト1　ファイルの内容を画面に表示(その1)

```
/*
     ファイルの内容を画面に表示する
*/
/***********************************************/
/** FILE 型を使うため stdio.h をインクルード **/
/** これで，NULL，EOF などの定数も使える       **/
/***********************************************/
#include <stdio.h>
main(argc,argv)
int argc;
char *argv[];
{
     FILE *fp;
     int ch;
     fp=fopen(argv[1],"r");          /* ファイルのオープン */
     while((ch=getc(fp)) != EOF)  /* 入力が尽きるまで読む */
         putchar(ch);               /* 読んだデータを書く */
     fclose(fp);                    /* ファイルのクローズ */
}
```

リスト2　ファイルの内容を画面に表示（その2）

```
/*
     ファイルの内容を画面に表示する
     （標準入力切り替え版）
*/
/***********************************************/
/** FILE 型を使うため stdio.h をインクルード **/
/** これで，NULL，EOF などの定数も使える       **/
/** stdin も使えるようになる                   **/
/***********************************************/
#include <stdio.h>
main(argc,argv)
int argc;
char *argv[];
{
     FILE *fp;
     int ch;
     fp=fopen(argv[1],"r");          /* ファイルのオープン */
     if(fp==NULL) fp=stdin;         /* オープンできなければ */
     while((ch=getc(fp)) != EOF)  /* 入力が尽きるまで読む */
         putchar(ch);               /* 読んだデータを書く */
     fclose(fp);                    /* ファイルのクローズ */
}
```

リスト3　ファイルの内容をコピー

```
/*
     ファイルの内容をコピーする
*/
/***********************************************/
/** FILE 型を使うため stdio.h をインクルード **/
/** これで，NULL，EOF などの定数も使える       **/
/***********************************************/
#include <stdio.h>
main(argc,argv)
int argc;
char *argv[];
{
     FILE *fpi,*fpo;
     int ch;
     fpi=fopen(argv[1],"rb");     /* 入力ファイルのオープン*/
     if(fpi==NULL){ /* オープンできないならエラー処理 */
         fprintf(stderr,"%sがオープンできません¥n",argv[1]);
         exit(1);    /* 強制終了 */
     }
     fpo=fopen(argv[2],"wb");     /* 出力ファイルのオープン*/
     if(fpo==NULL){ /* オープンできないならエラー処理 */
         fclose(fpi); /* 入力ファイルのクローズ */
         fprintf(stderr,"%sがオープンできません¥n",argv[2]);
         exit(1);        /* 強制終了 */
     }
     while((ch=getc(fpi)) != EOF) /* 入力が尽きるまで読む */
         putc(ch,fpo);                /* 読んだデータを書く */
     fclose(fpi);                     /*入力ファイルのクローズ*/
     fclose(fpo);                     /*出力ファイルのクローズ*/
}
```

い方，最後がほかのコマンドの標準出力をパイプで標準入力につなぎ替える使い方です。このプログラムの完全なリストをリスト2に示しておきます。実際に動作させて確かめてください。

さて，これまで出力ストリームは標準出力のみでしたが，こんどはファイルに出力する例を考えましょう。これは，FILE構造体へのポインタを入力用と出力用で2つ用意することで行うことができます。その例がリスト3のプログラムです。これはファイルの内容を別のファイルにコピーするプログラムです。リスト3では，テキストファイルでもバイナリファイルでもコピーできるように，ファイルの扱いをバイナリモードにしていますが，やっていることはリスト1，リスト2と大差ありません。出力先を，標準出力ではなくファイルに変更しただけというのがわかりますね。なお，リスト3ではファイルのオープンに失敗したときのエラー処理を追加してみました[12]。エラー処理の参考にしてください。

12) exit関数はオープンしているすべてのファイルをクローズするという副作用があるので，exit関数でプログラムの実行を終了する場合はすでにオープンしたファイルをクローズする必要はない。リスト3では出力ファイルのオープンが失敗すると，クローズすることを強調するために，わざわざ入力ファイルをクローズしている。

●1行単位の入出力

テキストファイルは1文字ずつ処理するよりも，1行ずつ処理するほうが都合のいいことがあります。Human68kのコマンドであるfind.xとかsort.xなどは行単位でファイルを処理するためのコマンドです。ここでは，1行単位のファイル処理の例として，ファイルに行番号を付けて，別のファイルに書き込むプログラムを作ってみます。それがリスト4のプログラムで，これは読み込んだテキストファイルに行番号を付けて出力するプログラムです。単純に考えれば読んだ1行を行番号を付けて出力するだけのプログラムになりますが，それだけでは面白くないので，リスト4にはちょっとした仕掛けがしてあります。

1行ずつ処理をするためには，1行分のデータを格納するためのchar型配列を用意しなければなりません。ところが，ファイルの1行の文字数はさまざまで予測がつきません。ときには1行分のデータが用意した配列に入り切らない場合もあります。1行ずつ処理を行うためには，このことを考慮しなければなりません。そもそも，fgets関数に読み込む最大文字数を指定するのは，配列の大きさを超えてデータを読み込むことを防ぐためなのです。リスト4は用意した文字列格納用の配列の大きさがなんであっても（ただし2バイト以上）正しい処理ができるようになっています[13]。いまは，MAXLINという定数で配列の大きさを指定しています（#defineというプリプロセッサの命令を使用しています）。この値を変更することで，いろいろな大きさの配列を指定することができるようになっているのです。

いま，リスト4のプログラムでは配列の大きさは，なんと2バイトしかありません。それでも，入力したテキストファイルに正しく行番号を付けることができるのです。この秘密を解説しましょう。それは，配列に読み込んだ最後の文字が改行文字かどうかを調べるところにあります。そして，改行文字を見つけない限りは行番号を書かないという制御をするのです。つまり，改行文字を読み込まないうちは1行分の出力が完結していないと考えて，行番号を書かずに読み込んだ内容を出力します。リスト4のpendingという変数が，改行文字を読み込んだかどうかの情報を保持しています。本来なら，プログラムの中心部を素直にプログラムすると，

cnt＝strlen(LINE);

リスト4　ファイルに行番号を付ける

```
/*
     ファイルに行番号を付ける

     用意した配列に1行が入り切らない場合も考慮する
*/
#include <stdio.h>
#define  MAXLIN 2    /* 配列の大きさを定数で与える */
char LINE[MAXLIN];
main(argc,argv)
int argc;
char    *argv[];
{
     FILE    *fpi,*fpo;
     int lnum=0;
     int pending=0;
     int cnt;
     fpi=fopen(argv[1],"r");      /* 入力ファイルをオープン*/
     if(fpi==NULL) fpi=stdin;     /* できなければ標準入力に*/
     fpo=fopen(argv[2],"w");      /* 出力ファイルをオープン*/
     if(fpo==NULL) fpo=stdout;    /* できなければ標準出力に*/
     while(fgets(LINE,MAXLIN,fpi)!=NULL){
          cnt=strlen(LINE);       /* 読んだ1行の長さ */
          if(pending==0)          /* 行が完結していれば行番号
を書く */
               fprintf(fpo,"%5d : ",++lnum);
          fputs(LINE,fpo);        /* 読んだ内容を書く */
          if(LINE[cnt-1]=='¥n')   /* 最後の文字が改行か？ */
               pending=0;         /* そうなら行は完結した */
          else
               pending=1;         /* 行は完結していない */
     }
     fclose(fpi);              /* 入力ファイルをクローズ */
     fclose(fpo);              /* 出力ファイルをクローズ */
}
```

リスト4の実行結果（自分自身を入力）

```
 1 : /*
 2 :      ファイルに行番号を付ける
 3 :
 4 :      用意した配列に1行が入り切らない場合も考慮する
 5 : */
 6 : #include <stdio.h>
 7 : #define  MAXLIN 2    /* 配列の大きさを定数で与える */
 8 : char LINE[MAXLIN];
 9 : main(argc,argv)
10 : int argc;
11 : char    *argv[];
12 : {
13 :      FILE    *fpi,*fpo;
14 :      int lnum=0;
15 :      int pending=0;
16 :      int cnt;
17 :      fpi=fopen(argv[1],"r");      /* 入力ファイルをオープン */
18 :      if(fpi==NULL) fpi=stdin;     /* できなければ標準入力に */
19 :      fpo=fopen(argv[2],"w");      /* 出力ファイルをオープン */
20 :      if(fpo==NULL) fpo=stdout;    /* できなければ標準出力に */
21 :      while(fgets(LINE,MAXLIN,fpi)!=NULL){
22 :           cnt=strlen(LINE);       /* 読んだ1行の長さ */
23 :           if(pending==0)          /* 行が完結していれば行番号を書く */
24 :                fprintf(fpo,"%5d : ",++lnum);
25 :           fputs(LINE,fpo);        /* 読んだ内容を書く */
26 :           if(LINE[cnt-1]=='¥n')   /* 最後の文字が改行か？ */
27 :                pending=0;         /* そうなら，行は完結した */
28 :           else
29 :                pending=1;         /* 行は完結していない */
30 :      }
31 :      fclose(fpi);              /* 入力ファイルをクローズ */
32 :      fclose(fpo);              /* 出力ファイルをクローズ */
33 : }
```

```
if(LINE [cnt−1]==’¥n’) {/＊最後が改行か？ ＊/
  if(pending) /＊行が完結していなければ＊/
    fprintf(fpo,”%s”,LINE);/＊行番号なし＊/
else/＊行が完結していれば，行番号あり＊/
fprintf(fpo,”%5d : %s”,++lnum,LINE);
pending=0；          /＊行は完結した＊/
}
else  {/＊最後の文字は改行ではない＊/
if(pending) /＊が完結していなければ＊/
  fprintf(fpo,”%s”,LINE);/＊行番号なし＊/
else  /＊行が完結していれば，行番号あり ＊/
  fprintf(fpo,”%5d : %s”,++lnum,LINE);
pending=1；          /＊行は完結していない＊/
}
```

となりますが，リスト4では最適化してあります。それ以外については特に説明する必要はないでしょう。

13) 配列の大きさが2バイトのときは，1文字処理の変形と見ることもできる。1行単位の処理は，結局1文字処理の特殊な場合なのかもしれない。

●1データ単位の入出力

今度は，ある程度まとまった大きさの単位でファイルの入出力を行う例を示しましょう。C言語でデータ処理を行う場合，データを配列なり構造体なりのまとまった単位で扱います。これらのデータをファイルから読んだりファイルに書き込む場合は，1バイトずつファイルを読み書きするよりも，意味的にまとまった単位で読み書きを行いたいものです。fread，fwriteという関数はそのような目的で入出力を行うための関数です。ここでは，第4回「配列って何だろう（1次元編）」や第5回「配列って何だろう(多次元編)」で例に取り上げた統計処理(試験の得点から偏差値を求めるというやつですね）のプログラムを，ファイルに対して行うことを考えます。このような処理は結果をファイルに保存したり，保存してあるファイルのデータをもとに別の処理を行うのが基本ですから，やっとこの手のプログラムの最終形に到達したことになります。リスト5がそのプログラムです。配列の説明をしたときは，データ構造を無理矢理配列に当てはめてプログラムを作ってみましたが，リスト5では，より自然に，構造体の配列を使ったデータ構造にしてあります。

プログラムの細かい説明はリストに書き込んだ注釈を見てもらうことにして，ここではプログラムの大まかな動作を説明します。リスト5のプログラムは，

(1) テキストファイル(input1.dat)に格納されている複数の組のデータを構造体の配列（RAW_DATA）に読み込む

(2) 構造体の配列の内容（RAW_DATA）で統計計算を行い，その処理結果を別の構造体の配列(NEW_DATA)に書き移す

(3) 統計計算を行った結果をデータファイル（output1.dat）に書き込む

(4) データファイル（output1.dat）の内容を再び構造体の配列（NEW_DATA）に読み込む

(5) 構造体の配列(NEW_DATA)の内容を処理する(クイックソート)

(6) 処理結果を別のデータファイル（output2.dat）に書き込む

という流れで処理されます。これを念頭においてリスト5のプログラムを読めば，やっていることはわかりますね。

さて，リスト5では（1）〜(3）と（4）〜(6）の処理を同時に行っていますが，アプリケーションの観点から見れば，本来は別のプログラムで行われるべき処理です。(1）〜(3）の処理は，（4）〜(6）で処理するためのデータファイル（output1.dat）を作るための処理で，いわば前処理です。リスト5のプログラムの本質は（4）〜(6）の部分にあります。つまり，リスト5はすでに存在しているデータファイルを加工し，結果を別のファイルとして保存することを目的としています。このようなファイル処理は，プログラミングの教育コースで行うファイル処理のもっとも基本的な部分なのです[14)]。

リスト5のプログラムで，データファイルにデータを読み書きするために使用している関数はfread，fwrite関数です。使用法を見てわかるように，これらの関数はファイルの内容を配列全体にデータを読み込んだり，配列全体のデータをファイルに書き込むのに適しています。1データ単位（配列要素）のバイト数とデータ数（配列の要素数）を引数で指定することになっているのはそのためなのです。そして，それさえ指定すれば，1回の関数呼び出しだけで，配列全体とファイルの間でデータの入れ替えができてしまうのですから，なかなか使いやすい関数といえます。あとは，リスト5のプログラムを実行してみて動きを追ってみてください。

14) C言語におけるファイル処理は，テキストファイルの変換にあると思う。つまり，テキストファイルの中で指定した文字列がある行を抜き出したり，ソースプログラムを実行形式のファイルに変換したり（アセンブラやコンパイラ）するのが通常の処理のように思える。データファイルを読み込み，統計処理をほどこして，別のファイルに保存するというファイル処理は，むしろBASIC，FORTRAN，COBOLなどの数値処理が得意な言語の守備範囲のような気がしないでもない。

●ランダムファイルの扱い

ファイルの内容は配列に読み込んで処理をするのがもっとも効率的です。配列に入れてしまえば，その任意の部分を添字で参照することができますから，ランダムアクセスを簡単に行うことができます。その意味で，ファイルの位置指示子を自由に移動するためのfseek関数は，メモリ容量に余裕のないときにファイルの内容をランダムアクセスするのにしか役に立たないような気がします。

実際のfseek関数の使用例としては，書き込みをしていたファイルを先頭から読み直すとか，ファイルのバイト数を知るといったものが考えられますが，あまりこれといった例を考えつきません（ちょっと高度な例は思いつくんだけどなあ)。いまのところはファイル位置指示子を変更できる関数があるということを覚えておけばよいでしょう。といってfseek関数の使用例を省略してしまうのも寂しいので，ごく簡単な例をプログラムしてみます。

リスト6のプログラムはリスト5のプログラムの変形です。プログラムの前半はリスト5そのもので，これはoutput1.datというデータファイルを作るためのものです。後半は，scanf関数で読み込む番号に対応するデータをoutput1.datの中から読み込んで画面に内容を表示するだけのプログラムです。scanf関数で読み込む値からファイルの先頭までのオフセットを計算し，その位置にfseek関数でファイル位置指示子を移動して，fread関数でひとつのデータを読み込むだけです。リスト5がわかっている人にはなんのこともないプログラムですね。なお，XCのVer.1.0を使っている人は（いないと思うけど)，fseek関数の第3引数のSEEK_SETは0に書き換えてください（stdio.hの中で定義されていない)。

ところで，リスト6のプログラムはscanf関数での入力に負数を与えると終了することになっています。このとき，数字以外の文字を入力するとscanf関数が暴走してしまうので注意しましょう。書式指定が"%d"なのでscanf関数が標準入力の中で整数値をどこまでも探し続けるためでしょう(詳しくはよくわかりません)。暴走が嫌な人はscanf関数を実行する部分のプログラムを次のように改造してみてください。

```
char tmp [10];
  ⋮
scanf("%s", tmp);
sscanf(tmp,"%d", &n);
```

つまり，標準入力をいったん文字列として，一時的な配列に取り込んでおき，それを改めてメモリ版のscanf関数（説明はしていませんが，ストリームの代わりに文字列からデータを取ってくるscanf関数です）であるsscanf関数で数値に変換します。こうすることで，scanf関数の暴走はなくなります。

◆基礎力を高めよう

設問1　fseek関数を使ってファイルの大きさ（バイト数）を知る方法を考えてください。

設問2　標準エラー出力(stderr)をファイルにリダイレクトするのは容易ではありません。しかし，freopen関数を使えば，プログラムで標準エラー出力をファイルに割り付けるのは難しくありません。その方法を考えてください。freopen関数については本文では説明していませんので，使い方はマニュアルで調べてください。

(解答は右)

おわりに

C言語のファイル入出力関数についてひととおりの説明をしてきましたが，理解できたでしょうか。これらの関数の使い方はそれほど難しくありません。要はそれをどういった目的に使用すればいいかということです。いまは何ができるのかよくわからないと思いますが，最初は標準入出力に対するデータをファイルに保存するというような使い方をしていけばよいでしょう。

さて，14回にわたって連載を続けてきたこの「ようこそここへC言語」は，今回で一応終了です。当初の目的はC言語のプログラムが書けるようになることでしたが，結果はどうでしょう。確かに，C言語の文法の説明にページを割きすぎて，実際のプログラム例が少なかったので，まだよくわからない人もいるかもしれません。それはひとえに私の力のなさに起因することです。ごめんなさい。しかし，これまでの連載で基本的な項目はすべて網羅してきたつもりですから，わかるようになるまで復習すれば，きっとわかるようになるでしょう（なんのこっちゃ)。

もし，ご要望があるようでしたら，第2部として実際のプログラムの作り方に関する連載を考えてはいます。それでは，ごきげんよう。

第 1 部 完

◆基礎力を高めようの解答

設問1

サイズを求めるファイルに対応するFILE構造体へのポインタをfpとすると，

```
fseek(fp, 0, SEEK_END);
```

または，

```
fseek(fp, 0, 2);
```

を実行したあと，

```
size=ftell(fp);
```

を実行することで，変数sizeにファイルのサイズを知ることができる。

解説

SEEK_ENDを基準にしてオフセット0の位置がファイルの最後になるので，そのときのファイル位置指示子を読み出せばよい。このときの値はテキストファイルでもバイナリファイルでも同じ値になる。

設問2

次のようにしてstderrを特定のファイルとしてfreopen関数でオープンし直す。

```
freopen("ERR", "w", stderr);
```

こうすることで，以後標準エラー出力(stderr)に書き込まれる内容は，freopen関数で指定したファイル（いまはERR）に書き込まれる。

解説

freopen関数はstdin，stdout，stderrなど標準的にオープンされるストリームを別のファイルに割り当てるのに使用する。freopen関数自身は再びオープンしたストリームに対応するFILE構造体へのポインタを返すが，エラーが発生でもしない限り，それは引数で指定したストリームと同じになる。したがって，freopen関数の戻り値は特に必要ない。また，freopen関数はオープンしているファイルの("r"とか"w"といった）アクセスモードを変更するのにも使用することができる。

リスト5　統計処理（その1）

```
/*
        構造体で統計的な処理を行うプログラム

*/

#include <stdio.h>
#include <math.h>

typedef struct {
    char    NAMAE[32];  /* 名前 */
    int EIGO;         /* 英語の得点 */
    int SUGAKU;       /* 数学の得点 */
    int KOKUGO;       /* 国語の得点 */
}   Mark;

typedef struct  {
    Mark    TOKUTEN;      /* 生の得点 */
    int SUM;              /* 3科目の得点合計 */
    int HENSA_EIGO;       /* 英語の偏差値 */
    int HENSA_SUGAKU;     /* 数学の偏差値 */
    int HENSA_KOKUGO;     /* 国語の偏差値 */
}   Personal;

Mark        RAW_DATA[50];   /* 生のデータ */
Personal    NEW_DATA[50];   /* 加工したデータ */

int     ndata;        /* データの個数 */

typedef struct {
    double  HEIKIN;       /* 平均値 */
    double  HYOJUN;       /* 標準偏差 */
    double  BUNSAN;       /* 分散 */
} BasData;

BasData EIGO_DAT,         /* 英語のデータ */
        SUGAKU_DAT,       /* 数学のデータ */
        KOKUGO_DAT;       /* 国語のデータ */

INPUT_T(fnam)  /** テキストデータの入力 **/
char    fnam[];
{
    FILE    *fp;
    Mark    *p= &RAW_DATA[0];

    ndata=0;
    if((fp=fopen(fnam, "r"))==NULL){
        fprintf(stderr,
            "%s 入力ファイルのオープンに失敗しました！¥n",
            fnam);
        exit(1);
    }
    while( fscanf(fp,"%s %d %d %d",
        p->NAMAE,
        &(p->EIGO),
        &(p->SUGAKU),
        &(p->KOKUGO) )!=EOF){
        ndata++;
        p++;
    }
    fclose(fp);
}

INPUT(fnam) /** バイナリデータの入力 **/
char    fnam[];
{
    FILE    *fp;

    if((fp=fopen(fnam, "rb"))==NULL){
        fprintf(stderr,
            "%s 入力ファイルのオープンに失敗しました！¥n",
            fnam);
        exit(1);
    }
    ndata=
        fread(NEW_DATA, sizeof(Personal), 50, fp);
    /* これだけ！ */
    fclose(fp);

}

OUTPUT(fnam)    /** バイナリデータの出力 **/
char    fnam[];
{
    FILE    *fp;

    if((fp=fopen(fnam, "wb"))==NULL){
        fprintf(stderr,
            "%s 出力ファイルのオープンに失敗しました！¥n",
            fnam);
        exit(1);
    }
    fwrite(NEW_DATA, sizeof(Personal), ndata, fp);
     /* これだけ！ */
    fclose(fp);

}

CLEAR() /** データ (NEW_DATA) を内容を無効化する関数 **/
{
    int i;
    for(i=0; i<ndata; i++)  /* 無意味なデータで書き潰す */
        strcpy(&NEW_DATA[i],"********************************
****");
}

CALC()  /** 統計処理を行う関数 **/
{
    int SUM_EIGO  =0,    /* 英語の得点の合計 */
        SUM_SUGAKU=0,    /* 数学の得点の合計 */
        SUM_KOKUGO=0;    /* 国語の得点の合計 */
    int i;

    /* 得点の合計を求める */

    for(i=0; i<ndata; i++ ){
        SUM_EIGO   += RAW_DATA[i].EIGO;
        SUM_SUGAKU += RAW_DATA[i].SUGAKU;
        SUM_KOKUGO += RAW_DATA[i].KOKUGO;
    }

    /* 各科目の得点の平均値を求める */

    EIGO_DAT.HEIKIN   = SUM_EIGO   / ndata;
    SUGAKU_DAT.HEIKIN = SUM_SUGAKU / ndata;
    KOKUGO_DAT.HEIKIN = SUM_KOKUGO / ndata;

    /* 各科目の得点の分散を求める */

    SUM_EIGO  =0;
    SUM_SUGAKU=0;
    SUM_KOKUGO=0;

    for(i=0; i<ndata; i++ ){     /* 偏差の平方和 */
     SUM_EIGO   += (RAW_DATA[i].EIGO   - EIGO_DAT.HEIKIN  )
            * ( RAW_DATA[i].EIGO   - EIGO_DAT.HEIKIN   );
     SUM_SUGAKU += (RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN)
            * ( RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN );
     SUM_KOKUGO += (RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN)
            * ( RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN );
    }

    EIGO_DAT.BUNSAN   = SUM_EIGO   / ndata;
    SUGAKU_DAT.BUNSAN = SUM_SUGAKU / ndata;
    KOKUGO_DAT.BUNSAN = SUM_KOKUGO / ndata;

    /* 各科目の得点の標準偏差を求める */

    EIGO_DAT.HYOJUN   = sqrt( EIGO_DAT.BUNSAN );
    SUGAKU_DAT.HYOJUN = sqrt( SUGAKU_DAT.BUNSAN );
    KOKUGO_DAT.HYOJUN = sqrt( KOKUGO_DAT.BUNSAN );

    /* 各自の偏差値を求める */

    for(i=0; i<ndata; i++ ){
        NEW_DATA[i].TOKUTEN = RAW_DATA[i];
        NEW_DATA[i].HENSA_EIGO=
            (RAW_DATA[i].EIGO   - EIGO_DAT.HEIKIN)*10/EIGO_
DAT.HYOJUN + 50;
        NEW_DATA[i].HENSA_SUGAKU=
            (RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN)*10/SUG
AKU_DAT.HYOJUN + 50;
        NEW_DATA[i].HENSA_KOKUGO=
            (RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN)*10/KOK
UGO_DAT.HYOJUN + 50;
    }

    /* 3科目の合計点を求める */

    for(i=0; i<ndata; i++)
        NEW_DATA[i].SUM =  RAW_DATA[i].EIGO
                + RAW_DATA[i].SUGAKU
                + RAW_DATA[i].KOKUGO;
}

REPORT() /* データ(NEW_DATA)の内容を画面(標準出力)に書く */
{
    int i;

    for(i=0; i<ndata ; i++){
     printf("%4d %-14s¥t",(i+1),NEW_DATA[i].TOKUTEN.NAMAE);
     printf("英語 %3d (%3d)¥t", NEW_DATA[i].TOKUTEN.EIGO,
                NEW_DATA[i].HENSA_EIGO);
     printf("数学 %3d (%3d)¥t", NEW_DATA[i].TOKUTEN.SUGAKU,
                NEW_DATA[i].HENSA_SUGAKU);
     printf("国語 %3d (%3d)¥t", NEW_DATA[i].TOKUTEN.KOKUGO,
                NEW_DATA[i].HENSA_KOKUGO);
     printf("合計 %3d¥n",        NEW_DATA[i].SUM);
    }
    printf("¥n英語 平均 %5.1f : 標準偏差 %5.1f¥n",
                EIGO_DAT.HEIKIN, EIGO_DAT.HYOJUN);
    printf("数学 平均 %5.1f : 標準偏差 %5.1f¥n",
                SUGAKU_DAT.HEIKIN, SUGAKU_DAT.HYOJUN);
    printf("国語 平均 %5.1f : 標準偏差 %5.1f¥n¥n¥n",
                KOKUGO_DAT.HEIKIN, KOKUGO_DAT.HYOJUN);
}

comp(x,y)   /** クイックソート用，比較関数（降順） **/
Personal    *x,*y;
{
    if( x->SUM < y->SUM ) return 1;
    if( x->SUM > y->SUM ) return -1;
    return 0;
}

main()
{
    INPUT_T("input1.dat");  /* データの入力（テキストファイ
ルから） */
    CALC();                 /* 統計計算 */
    REPORT();               /* 結果を画面にプリント */
    OUTPUT("output1.dat");  /* データの出力 */
    CLEAR();                /* データを消去（念のため） */
    INPUT("output1.dat");   /* データの入力（データファイル
から） */

    qsort(NEW_DATA, ndata, sizeof(Personal), (int (*)())com
p);  /* ソートだ！ */

    OUTPUT("output2.dat");  /* データの出力 */
    REPORT();
}
```

リスト5の実行結果

```
(1)入力ファイル(input1.dat)の例

        陸奥_九十九      90      65      80
        龍造寺_舞子      60     100      55
        海堂_晃          75      90      80
        中山_美穂        80      89      71
        増畑_大志        90      75      60
        泉_敏彦          88      82      65
        陣雷_浩一        51      75      90
        南野_陽子        15      80      85
        片山_右京        30      56      94
        後藤_久美子      60      73      98
        酒井_法子        81      85      77
        奥寺_鉄二        77     100      67
        飛田_高明        68      87      80
        菊池_桃子        65      72      48
        羽山_悟          93      88      76
        宮沢_りえ        88      59      58
        杉本_彩          45      60      86
        竹海_直人        78      77      58
        不破_北斗        40      80      60

(2)出力ファイル(output1.dat)のダンプリスト,dump.xの結果(1部)

00000000  97 A4 89 9C 5F 8B E3 8F 5C 8B E3 00 00 00 00 00 陸奥_九十九.....
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000020  00 00 00 5A 00 00 00 41 00 00 00 50 00 00 00 EB ...Z...A...P....
00000030  00 00 00 3C 00 00 00 27 00 00 00 37 97 B4 91 A2 ...<...'...7龍造
00000040  8E 9B 5F 95 91 8E 71 00 00 00 00 00 00 00 00 00 寺_舞子.........
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3C ...............<
00000060  00 00 00 64 00 00 00 37 00 00 00 D7 00 00 00 2E ...d...7...ﾗ....
00000070  00 00 00 43 00 00 00 25 8A 43 93 B0 5F 8D 57 00 ...C...%海堂_晃.
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000090  00 00 00 00 00 00 00 00 00 00 00 4B 00 00 00 5A ...........K...Z
000000A0  00 00 00 50 00 00 00 F5 00 00 00 35 00 00 00 3B ...P...   ..5...;
000000B0  00 00 00 37 92 86 8E 52 5F 94 FC 95 E4 00 00 00 ...7中山_美穂...
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000D0  00 00 00 00 00 00 00 50 00 00 00 59 00 00 00 47 .......P...Y...G
000000E0  00 00 00 F0 00 00 00 38 00 00 00 3A 00 00 00 30 ...

(3)画面(標準出力)への表示結果

    1  陸奥_九十九      英語  90 ( 60)  数学  65 ( 39)  国語  80 ( 55)  合計 235
    2  龍造寺_舞子      英語  60 ( 46)  数学 100 ( 67)  国語  55 ( 37)  合計 215
    3  海堂_晃          英語  75 ( 53)  数学  90 ( 59)  国語  80 ( 55)  合計 245
    4  中山_美穂        英語  80 ( 56)  数学  89 ( 58)  国語  71 ( 48)  合計 240
    5  増畑_大志        英語  90 ( 60)  数学  75 ( 47)  国語  60 ( 40)  合計 225
    6  泉_敏彦          英語  88 ( 59)  数学  82 ( 53)  国語  65 ( 44)  合計 235
    7  陣雷_浩一        英語  51 ( 42)  数学  75 ( 47)  国語  90 ( 62)  合計 216
    8  南野_陽子        英語  15 ( 25)  数学  80 ( 51)  国語  85 ( 58)  合計 180
    9  片山_右京        英語  30 ( 32)  数学  56 ( 32)  国語  94 ( 65)  合計 180
   10  後藤_久美子      英語  60 ( 46)  数学  73 ( 45)  国語  98 ( 67)  合計 231
   11  酒井_法子        英語  81 ( 56)  数学  85 ( 55)  国語  77 ( 52)  合計 243
   12  奥寺_鉄二        英語  77 ( 54)  数学 100 ( 67)  国語  67 ( 45)  合計 244
   13  飛田_高明        英語  68 ( 50)  数学  87 ( 57)  国語  80 ( 55)  合計 235
   14  菊池_桃子        英語  65 ( 49)  数学  72 ( 45)  国語  48 ( 32)  合計 185
   15  羽山_悟          英語  93 ( 62)  数学  88 ( 58)  国語  76 ( 52)  合計 257
   16  宮沢_りえ        英語  88 ( 59)  数学  59 ( 34)  国語  58 ( 39)  合計 205
   17  杉本_彩          英語  45 ( 39)  数学  60 ( 35)  国語  86 ( 59)  合計 191
   18  竹海_直人        英語  78 ( 55)  数学  77 ( 49)  国語  58 ( 39)  合計 213
   19  不破_北斗        英語  40 ( 37)  数学  80 ( 51)  国語  60 ( 40)  合計 180

英語 平均  67.0 : 標準偏差  21.7
数学 平均  78.0 : 標準偏差  12.3
国語 平均  73.0 : 標準偏差  13.9

    1  羽山_悟          英語  93 ( 62)  数学  88 ( 58)  国語  76 ( 52)  合計 257
    2  海堂_晃          英語  75 ( 53)  数学  90 ( 59)  国語  80 ( 55)  合計 245
    3  奥寺_鉄二        英語  77 ( 54)  数学 100 ( 67)  国語  67 ( 45)  合計 244
    4  酒井_法子        英語  81 ( 56)  数学  85 ( 55)  国語  77 ( 52)  合計 243
    5  中山_美穂        英語  80 ( 56)  数学  89 ( 58)  国語  71 ( 48)  合計 240
    6  泉_敏彦          英語  88 ( 59)  数学  82 ( 53)  国語  65 ( 44)  合計 235
    7  陸奥_九十九      英語  90 ( 60)  数学  65 ( 39)  国語  80 ( 55)  合計 235
    8  飛田_高明        英語  68 ( 50)  数学  87 ( 57)  国語  80 ( 55)  合計 235
    9  後藤_久美子      英語  60 ( 46)  数学  73 ( 45)  国語  98 ( 67)  合計 231
   10  増畑_大志        英語  90 ( 60)  数学  75 ( 47)  国語  60 ( 40)  合計 225
   11  陣雷_浩一        英語  51 ( 42)  数学  75 ( 47)  国語  90 ( 62)  合計 216
   12  龍造寺_舞子      英語  60 ( 46)  数学 100 ( 67)  国語  55 ( 37)  合計 215
   13  竹海_直人        英語  78 ( 55)  数学  77 ( 49)  国語  58 ( 39)  合計 213
   14  宮沢_りえ        英語  88 ( 59)  数学  59 ( 34)  国語  58 ( 39)  合計 205
   15  杉本_彩          英語  45 ( 39)  数学  60 ( 35)  国語  86 ( 59)  合計 191
   16  菊池_桃子        英語  65 ( 49)  数学  72 ( 45)  国語  48 ( 32)  合計 185
   17  南野_陽子        英語  15 ( 25)  数学  80 ( 51)  国語  85 ( 58)  合計 180
   18  片山_右京        英語  30 ( 32)  数学  56 ( 32)  国語  94 ( 65)  合計 180
   19  不破_北斗        英語  40 ( 37)  数学  80 ( 51)  国語  60 ( 40)  合計 180

英語 平均  67.0 : 標準偏差  21.7
数学 平均  78.0 : 標準偏差  12.3
国語 平均  73.0 : 標準偏差  13.9
```

リスト6　統計処理(その2)

```
/*
        構造体で統計的な処理を行うプログラム

        番号を指定して,結果をランダムに参照する
*/

#include <stdio.h>
#include <math.h>

typedef struct {
    char    NAMAE[32];   /* 名前 */
    int EIGO;            /* 英語の得点 */
    int SUGAKU;          /* 数学の得点 */
    int KOKUGO;          /* 国語の得点 */
```

▶アクアレスの「うりうり」が,なかなか卑怯くさくて気持ちよかった。エグザクトの人たちには過労死しないていどでがんばってもらいましょう。　山中　一男(17)栃木県

```
} Mark;

typedef struct {
    Mark    TOKUTEN;        /* 生の得点 */
    int SUM;                /* 3科目の得点合計 */
    int HENSA_EIGO;         /* 英語の偏差値 */
    int HENSA_SUGAKU;       /* 数学の偏差値 */
    int HENSA_KOKUGO;       /* 国語の偏差値 */
} Personal;

Mark        RAW_DATA[50];   /* 生のデータ */
Personal    NEW_DATA[50];   /* 加工したデータ */

int     ndata;      /* データの個数 */

typedef struct {
    double  HEIKIN;     /* 平均値 */
    double  HYOJUN;     /* 標準偏差 */
    double  BUNSAN;     /* 分散 */
} BasData;

BasData EIGO_DAT,       /* 英語のデータ */
        SUGAKU_DAT,     /* 数学のデータ */
        KOKUGO_DAT;     /* 国語のデータ */

INPUT_T(fnam)   /** テキストデータの入力 **/
char    fnam[];
{
    FILE    *fp;
    Mark    *p= &RAW_DATA[0];

    ndata=0;
    if((fp=fopen(fnam, "r"))==NULL){
        fprintf(stderr,
            "%s 入力ファイルのオープンに失敗しました！¥n",
            fnam);
        exit(1);
    }
    while( fscanf(fp,"%s %d %d %d",
        p->NAMAE,
        &(p->EIGO),
        &(p->SUGAKU),
        &(p->KOKUGO) )!=EOF){
        ndata++;
        p++;
    }
    fclose(fp);
}

OUTPUT(fnam)    /** バイナリデータの出力 **/
char    fnam[];
{
    FILE    *fp;

    if((fp=fopen(fnam, "wb"))==NULL){
        fprintf(stderr,
            "%s 出力ファイルのオープンに失敗しました！¥n",
            fnam);
        exit(1);
    }
    fwrite(NEW_DATA, sizeof(Personal), ndata, fp);
    /* これだけ！ */
    fclose(fp);

}

CALC()  /** 統計処理を行う関数 **/
{
    int SUM_EIGO  =0,   /* 英語の得点の合計 */
        SUM_SUGAKU=0,   /* 数学の得点の合計 */
        SUM_KOKUGO=0;   /* 国語の得点の合計 */
    int i;

    /* 得点の合計を求める */

    for(i=0; i<ndata; i++ ){
        SUM_EIGO   += RAW_DATA[i].EIGO;
        SUM_SUGAKU += RAW_DATA[i].SUGAKU;
        SUM_KOKUGO += RAW_DATA[i].KOKUGO;
    }

    /* 各科目の得点の平均値を求める */

    EIGO_DAT.HEIKIN   = SUM_EIGO   / ndata;
    SUGAKU_DAT.HEIKIN = SUM_SUGAKU / ndata;
    KOKUGO_DAT.HEIKIN = SUM_KOKUGO / ndata;

    /* 各科目の得点の分散を求める */

    SUM_EIGO  =0;
    SUM_SUGAKU=0;
    SUM_KOKUGO=0;

  for(i=0; i<ndata; i++ ){     /* 偏差の平方和 */
   SUM_EIGO   += ( RAW_DATA[i].EIGO   - EIGO_DAT.HEIKIN  )
               * ( RAW_DATA[i].EIGO   - EIGO_DAT.HEIKIN  );
   SUM_SUGAKU += ( RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN)
               * ( RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN);
   SUM_KOKUGO += ( RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN)
               * ( RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN);
    }

    EIGO_DAT.BUNSAN   = SUM_EIGO   / ndata;
    SUGAKU_DAT.BUNSAN = SUM_SUGAKU / ndata;
    KOKUGO_DAT.BUNSAN = SUM_KOKUGO / ndata;

    /* 各科目の得点の標準偏差を求める */

    EIGO_DAT.HYOJUN   = sqrt( EIGO_DAT.BUNSAN );
    SUGAKU_DAT.HYOJUN = sqrt( SUGAKU_DAT.BUNSAN );
    KOKUGO_DAT.HYOJUN = sqrt( KOKUGO_DAT.BUNSAN );

    /* 各自の偏差値を求める */

    for(i=0; i<ndata; i++ ){
        NEW_DATA[i].TOKUTEN = RAW_DATA[i];
        NEW_DATA[i].HENSA_EIGO=
            (RAW_DATA[i].EIGO    - EIGO_DAT.HEIKIN)*10/EIGO_
DAT.HYOJUN + 50;
        NEW_DATA[i].HENSA_SUGAKU=
            (RAW_DATA[i].SUGAKU - SUGAKU_DAT.HEIKIN)*10/SUG
AKU_DAT.HYOJUN + 50;
        NEW_DATA[i].HENSA_KOKUGO=
            (RAW_DATA[i].KOKUGO - KOKUGO_DAT.HEIKIN)*10/KOK
UGO_DAT.HYOJUN + 50;
    }

    /* 3科目の合計点を求める */

    for(i=0; i<ndata; i++)
        NEW_DATA[i].SUM =  RAW_DATA[i].EIGO
                   + RAW_DATA[i].SUGAKU
                   + RAW_DATA[i].KOKUGO;
}

main()
{
    FILE    *fp;
    int     n;
    int     offset;
    Personal pdata;

    INPUT_T("input1.dat");  /* データの入力
                               (テキストファイルから) */
    CALC();                 /* 統計計算 */
    OUTPUT("output1.dat");  /* データの出力 */

    if((fp=fopen("output1.dat", "r+b"))==NULL){
                                    /* 読み書きモード */
        fprintf(stderr,"入力ファイルのオープンに失敗しまし
た！¥n");
        exit(1);
    }

    while(1){
        printf("データの番号？");
        scanf("%d", &n);
        if( n < 0 ) break;  /* 負数を入力すると終わり */
        if( n >= ndata ){
            printf("番号が不正です¥n");
            continue;
        }
        offset = n*sizeof(Personal); /* オフセットの計算 */
        fseek(fp, offset, SEEK_SET); /* 先頭を基準にファイ
ル位置指示子を移動 */
        fread(&pdata, sizeof(Personal), 1, fp);
    /* 1データ読み込み  */

        printf("%4d  %-14s¥t",  n, pdata.TOKUTEN.NAMAE);
  /* データを表示 */
        printf("英語 %3d (%3d)¥t", pdata.TOKUTEN.EIGO,
 pdata.HENSA_EIGO);
        printf("数学 %3d (%3d)¥t", pdata.TOKUTEN.SUGAKU,
 pdata.HENSA_SUGAKU);
        printf("国語 %3d (%3d)¥t", pdata.TOKUTEN.KOKUGO,
 pdata.HENSA_KOKUGO);
        printf("合計 %3d¥n",       pdata.SUM);
    }

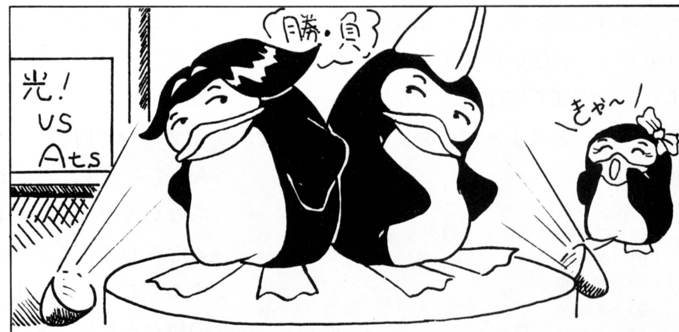
    fclose(fp);             /* クローズ */
}
```

リスト6の実行結果

```
データの番号？                ←  3を入力
   3  中山_美穂        英語  80 ( 56)  数学  89 ( 58)  国語  71 ( 48)  合計 240
データの番号？                ←  1を入力
   1  龍造寺_舞子      英語  60 ( 46)  数学 100 ( 67)  国語  55 ( 37)  合計 215
データの番号？                ←  14を入力
  14  羽山_悟          英語  93 ( 62)  数学  88 ( 58)  国語  76 ( 52)  合計 257
データの番号？                ←  －1を入力
```

マシン語カクテル in Z80's Bar

第27回 –炎のプログラミング勝負–



シナリオ：**柴田 淳**

前回から新メンバーになった柴田君と常連の１番手の光君がプログラムで対決。さて，軍配はどちらに上がるか。ツケがかかっているとあって，両者ともに戦いの火花を散らすつもりだったようですが，結局２人で花火を上げたようです。

柴田淳（以下**Ats**）：この間，学校の帰りに電車に乗ってたんですよ。

ようこ（以下**Yo**）：そりゃ，乗るでしょうね。

Ats：そしたら，いきなり鼻血が出てきちゃって。

Yo：鼻血？

Ats：最初は鼻水かと思ってたんですよ。でも，鼻水にしては粘り気が足りなくて，吸っても吸っても垂れてくるんですよ。で，とうとうぽたぽた落ちてきて，ズボンを汚しちゃまずいから……。

Yo：椅子に座ってたわけね。

Ats：そうなんですよ。でね，しょうがないから手で受けてたんですよ。

Yo：ポケットティッシュかなんか持ってなかったの？

Ats：うーん，見るに見かねた隣のおばさんが貸してくれたんで，その場は収まったんですけどね。周りの人にはじろじろ見られるし，まったく散々でしたよ。

Yo：やらしいこと考えてたんでしょ。

Ats：違いますよ。だいたいやらしいこと考えて鼻血が出るなんて，そんなことあるわけないじゃないですか。

Yo：そうかしらねえ。おおかた，目の前にミニスカートの可愛い女の子でもいたんじゃない？

Ats：……はあ。なんだかなあ。僕ってけっこう情けないよなあ。

Yo：まあ，そんなに思い詰めないほうがいいわよ。上には上があるっていうし。

Ats：……。光君，早く来ないかなあ。

♪ガラーン，ゴローン

源光(以下**光**)：こんばんは。あっ，まだドアベル直ってない。

Yo：光君いらっしゃい。

Ats：それと同じドアベル，なかなか見つからなくてさあ。

光：柴田君来てたんだ。

Ats：こんにちは。

光：……。なに２人して，枯れた老人のような顔してるんですか。

Yo：光君のこと待ってたのに，そのいい方はないんじゃない？

光：待ってた？ 僕のことを？ そういえばマスターがいないようだけど。

Ats：商店会の集まりだって。

Yo：それでね，マスターが出ている間に２人にやってもらいたいことがあるの。ちなみにスポンサーはマスター。

光：２人って，僕と柴田君のこと？

Yo：決まってるじゃない。同じサブルーチンを使って，２人にそれぞれプログラムを作ってもらいたいの。

Ats：そして出来具合によって，勝敗を決めるんだって。

Yo：審判はもちろんわたしね。

光：ちょっと待ってくださいよ。僕はいやですからね。第一，面倒臭いじゃないですか。ツケがたまってるからと脅かされたって，今回だけは絶対いやですからね。

Yo：まったくせっかちなんだから。話は最後まで聞きなさいよ。プログラムを作ってもらって，勝負に勝ったほうが……。

Ats：なんと，ツケを半分にしてもらえるんだって。

光：ツケ半分！ やるやる！ いや，やらせろ，じゃなかった，ぜひ，やらせてください！

 属性テーブルって何だ

光：同じサブルーチンって，何のサブルーチンを使うんですか？

Yo：待ってる間に柴田君が作ってくれたんだけど。

Ats：ええと，座標を動かすサブルーチンとか，テーブルのイニシャライズルーチンとか，乱数発生とか，まあそんなもんかな。これがリスト(リスト１)。コメントも書いておいたから，見ればだいたいわかるでしょ。

光：なるほど，よくあるやつね。

Ats：属性テーブルは16バイト長で，X，Yの増分と，スピードが別々に設定できるようになってる。

光：この方向っていうのは？

Ats：ああ，ビット０とビット１がそれぞれXとYに対応していて，ビットが立っていればプラス。

光：立っていなければマイナスでしょ。これはちょっと面倒臭そうだな。

Ats：まあ，方法はいろいろあるんだろうけど。あと，10番目から先は使ってなくて，スピードは16段階ね。僕のはもう出来上がってるから。

光：どんなやつ作ったの？

Ats：それは教えられないよ。それと，テーブルのスタートアドレスは7100$_H$ね。

光：ふーん。とにかくやってみようかな。１時間くらいでできると思うけど。ちょっと失礼。

Yo：あれ，これだけでわかっちゃうの？

Ats：サブルーチンっていうのは，パラメ

ータの渡し方とか，どんな機能があるかだけを知ってればいいですからね。使う側にすれば，内部で実際にどんな処理をしているかってことはまったく関係ないんですよ。

表1

00	ステータス　　0以外なら使用中
01	X座標
02	Y座標
03	X増分
04	Y増分
05	スピード
06	方向　0左上　1右上　2左下　3右下
07	X増分用カウンタ
08	Y増分用カウンタ
09	スピード用カウンタ
0A	未使用
0B	未使用
0C	未使用
0D	未使用
0E	未使用
0F	未使用

ようこさんも使い方くらいはわかったでしょ？

Yo：ぜんぜんわかんない。だいたい属性テーブルって何？

Ats：そうか，そこから説明しなくちゃならないんだ。たとえばシューティングゲームなんかでたくさんの敵キャラを動かすとするでしょ。するとキャラの数だけ，座標の値なんかを，どこかに置いておかなくちゃならないじゃないですか。

Yo：BASICとかCだったら配列を用意すればいいわよね。

Ats：でも，マシン語の場合はそうはいかなくて，直接メモリに書き込まなくちゃならない。それにひとつのサブルーチンだけがその値を参照したり書き換えたりするってことは，まずありえないですよね。

Yo：そうよね。最低限，表示するサブルーチンと移動させるサブルーチンが必要ね。

Ats：だからプログラムを作る前に，あらかじめどこにどの値が書き込まれているかっていうことを決めておかなくちゃならないでしょ。えーと，光君，さっき渡した表を貸してくれないかなあ（表1）。

光：ああ，これね。使うから早く返して。

Yo：ふーん，X座標とかY座標とかはわかるけど，X増分とか，方向って何？

Ats：ええと，順番に説明しますね。X増分っていうのは，X座標の増分のことで。

Yo：そのまんまじゃない。

Ats：いや，だからX座標の値が増える分ってことで。

Yo：X方向のスピードってことでしょ。そうか，さっきスピードは16段階っていって

リスト1

```
0000              ; ##-------------------##
0000              ; ##    SUB LOUTINES    ##
0000              ; ##-------------------##
0000              ;               PLOGRAM (ats)
0000              ;
7000              START  $7000
7000              ;
7000              ; ## MOVING CHARACTER ##
7000              ;
7000              ;          IX <-- STATUS ADD.
7000             #MOVER
7000 C5           PUSH   BC
7001              ;  MANAGING SPEED
7001              ;
7001 DD 7E 05     LD     A,(IX+5)
7004 DD 86 09     ADD    A,(IX+9)
7007 47           LD     B,A
7008 E6 0F        AND    15
700A DD 77 09     LD     (IX+9),A
700D B8           CP     B
700E CA 50 70     JP     Z,#RET
7011              ;  MANAGING -X-
7011              ;
7011 DD 4E 06     LD     C,(IX+6)
7014 DD 7E 03     LD     A,(IX+3)
7017 DD 86 07     ADD    A,(IX+7)
701A 47           LD     B,A
701B E6 0F        AND    15
701D DD 77 07     LD     (IX+7),A
7020 B8           CP     B
7021 CA 32 70     JP     Z,#YMNG
7024 CB 41        BIT    0,C
7026 CA 2F 70     JP     Z,#XMINUS
7029 DD 34 01     INC    (IX+1)
702C C3 32 70     JP     #YMNG
702F             #XMINUS
702F DD 35 01     DEC    (IX+1)
7032              ;  MANAGING -Y-
7032              ;
7032             #YMNG
7032 DD 7E 04     LD     A,(IX+4)
7035 DD 86 08     ADD    A,(IX+8)
7038 47           LD     B,A
7039 E6 0F        AND    15
703B DD 77 08     LD     (IX+8),A
703E B8           CP     B
703F CA 50 70     JP     Z,#RET
7042 CB 49        BIT    1,C
7044 CA 4D 70     JP     Z,#YMINUS
7047 DD 34 02     INC    (IX+2)
704A C3 50 70     JP     #RET
704D             #YMINUS
704D DD 35 02     DEC    (IX+2)
7050             #RET
7050 C1           POP    BC
7051 C9           RET
7052              ; ##  SEARCHING BLANK  ##
7052              ;          HL <-- START ADD.
7052              ;          B  <-- TIMES
7052             #?BLANK
7052 D5           PUSH   DE
7053 C5           PUSH   BC
7054 11 10 00     LD     DE,16
7057 3E 00        LD     A,0
7059             #LOOPBL
7059 BE           CP     (HL)
705A CA 61 70     JP     Z,#FIND
705D 19           ADD    HL,DE
705E 10 F9        DJNZ   #LOOPBL
7060 37           SCF
7061             #FIND
7061 C1           POP    BC
7062 D1           POP    DE
7063 C9           RET
7064              ; ###   INITIALIZE    ###
7064              ;          HL <-- START ADD.
7064              ;          B  <-- TIMES
7064             #INIT
7064 E5           PUSH   HL
7065 C5           PUSH   BC
7066 3E 00        LD     A,0
7068             #LOOPIT1
7068 0E 10        LD     C,16
706A             #LOOPIT2
706A 77           LD     (HL),A
706B 23           INC    HL
706C 0D           DEC    C
706D C2 6A 70     JP     NZ,#LOOPIT2
7070 10 F6        DJNZ   #LOOPIT1
7072 C1           POP    BC
7073 E1           POP    HL
7074 C9           RET
7075              ; ###  RANDOM VER. ###
7075              ;
7075              ;        A  <-- RANDOM VER
7075             #RND
7075 C5           PUSH   BC
7076 D5           PUSH   DE
7077 E5           PUSH   HL
7078 21 A4 70     LD     HL,#RNDBIT1
707B CD 9A 70     CALL   #RND2
707E F5           PUSH   AF
707F D1           POP    DE
7080 21 A5 70     LD     HL,#RNDBIT2
7083 CD 9A 70     CALL   #RND2
7086 F5           PUSH   AF
7087 C1           POP    BC
7088 79           LD     A,C
7089 AB           XOR    E
708A 1F           RRA
708B 2A A2 70     LD     HL,(#RNDBUFF)
708E ED 6A        ADC    HL,HL
7090 22 A2 70     LD     (#RNDBUFF),HL
7093 E1           POP    HL
7094 D1           POP    DE
7095 C1           POP    BC
7096 3A A2 70     LD     A,(#RNDBUFF)
7099 C9           RET
709A             #RND2
709A 46           LD     B,(HL)
709B 2A A2 70     LD     HL,(#RNDBUFF)
709E             #RNDLOOP
709E 29           ADD    HL,HL
709F 10 FD        DJNZ   #RNDLOOP
70A1 C9           RET
70A2             #RNDBUFF
70A2 14 56                DW $5614
70A4             #RNDBIT1
70A4 10                   DB 16
70A5             #RNDBIT2
70A5 02                   DB 2
70A6              ; ###      WAIT      ###
70A6              ;
70A6              ;        B  <-- TIMES
70A6             #WAIT
70A6 C5           PUSH   BC
70A7             #LOOPWT
70A7 C5           PUSH   BC
70A8 C1           POP    BC
70A9 10 FC        DJNZ   #LOOPWT
70AB C1           POP    BC
70AC C9           RET
OBJECT CODE END 70AC
```

たのはこのことなのね。すると次のY増分もスピードも同じようなものね。あれ，でもX座標もY座標も，増えるばっかりじゃだめなんじゃない？

Ats：おっ，するどいですね。そのために次の方向っていうパラメーターがあるんですよ。XとYにそれぞれ増やすか減らすかを設定するから，４方向ってことになるでしょ。さっき光君と話してたとき「ビット０とビット１がそれぞれXとYに対応してて」っていったでしょ。

Yo：ほうほう，なるほどね。それと，X，Yの増分とスピードが別々に設定できるっていうのはどうして？　X，Yの増分っていうのはその方向のスピードってことでしょ。もうひとつスピードがあるなんておかしいじゃない。

Ats：だんだん話がこんがらがってきたなあ。光君，リストと表を交換しよう。

光：……。

Yo：なんか燃えちゃってるわよ。

Ats：やっぱりお金がかかってるからな。そーっと入れ替えとこう。



スピード16分割

Ats：えーと，なんでX，Yの増分があるのに，それとは別にスピードを設定する必要があるのかってことですけど。

Yo：これが柴田君がさっき作ってたサブルーチンね。けっこう短いじゃない（リスト１）。

Ats：たいしたことやってるわけじゃないですからね。

Yo：でも，なんでリストなんか持ち出してきたの？

Ats：いや，プログラムを説明したほうが手っ取り早いと思ったんです。11行からの#MOVERっていうルーチンが，さっきいったテーブルの値を参照して座標を動かすためのものなんですけど，15行から先を見てください。

Yo：IXレジスタにテーブルの先頭番地を入れてコールするのね。15行でテーブルの先頭番地＋５の値をAに読み込むでしょ。５番目って何の値だったっけ？

Ats：スピード。

Yo：そのあとでスピードと先頭＋９の値を足してるけど，これはなんで？

Ats：先頭＋９番目は，スピード用のカウンタとして使ってるんですよ。足したあと，17行から21行でやってる処理がこのルーチンのミソなんですけど，何やってるかわかりますか，ようこさん？

Yo：……わかんない。

Ats：簡単にいえば，スピードを足したカウンタの値が16以上であればカウンタから16を引いて，次のXとYの値を動かす処理に飛んで，16以下だったらそのまま何もせずにリターンするっていう処理をしてるんですけどね。

Yo：ねえ，ここって速度を変える処理をしてるところでしょ。これといまいったことと何か関係があるの？

Ats：そうくると思った。いいですか，たとえばIX＋５の値が７だったとしますよ。最初にここを通ると，カウンタの値は７になりますよね。だからX，Yの値は変わらないでしょ。２回目は14になる。

Yo：え，それじゃスピードが７のときは，２回とも何もしないで呼び出したルーチンに帰っちゃうの？

Ats：そうなんですよ。３回目にカウンタが21になるから，やっと次のX，Yの値を変える処理を通る。

Yo：ああ，やっとわかった。スピードを変えるっていうから，値が高くなるほど一度にたくさん動かすんだと思ってた。そうか，呼び出すたびに動かすのが最高速で，それからスピードが下がるにつれて，だんだん遅くしていくのね，この場合は。なるほどなるほど。

Ats：やっとわかってもらえたか。

Yo：でもちょっと待ってよ。７っていったらスピードが８のときより16分の１だけ遅いんでしょ。でもさっきのだと３回に１回しか座標を動かしてないじゃない。８は16の半分，２回に１回の割合で座標を動かすんだから，16分の１どころか，もっと遅くなってない？

Ats：そんなことないですよ。さっきの続きをやってみましょう。カウンタが16以上になったら，16を引くんでしたよね。

Yo：ええと，21引く16は５でしょ。４回目は５に７を足して12，12に７を足して19。

Ats：ほらね。今度は２回に１回になったでしょ。

Yo：あらほんとだ。なんだか不思議ね。



Ats：カウンタが16を超えたら，ゼロにするんじゃなくて16を引くってところがポイントなんですよ。つまり，はみ出した分を取っておいて，次回にまわすってわけですね。。

Yo：ふーん。じゃあ，そのあとのXとYの値を変える部分でも，同じようなことをやってるんでしょ。

Ats：IX＋７と８をそのためのカウンタとして使ってます。

Yo：これでスピードに合わせて座標を動かす仕組みはわかったけど，なんでX，Yの増分があるのにわざわざ別にスピードのパラメータを用意したのか，っていう質問には答えてないじゃない。

Ats：そういえばそうですね。XYの増分のほかにスピードを設けたのはなぜかというと。

Yo：なぜかというと？

Ats：そのほうが便利だから。

Yo：え？

Ats：だから，そうしたほうが便利だからです。

Yo：そのまんまじゃない！

Ats：……。ようこさん，その漫才師みたいなきつい突っ込みはやめてくださいよ。だいたい古いですよ，そんなの。

Yo：……ごめん。



星に願いを？

光：やった！ 完成だ！

Ats：お，光君のができたみたいですよ。

Yo：ほんとに１時間で作っちゃったわね。ところでどっちのから先に見るかだけど，できた順でいえば最初は柴田君よね。

光：そうだね。僕もあとのほうがいいや。

Ats：それじゃあちょっと失礼。まずさっきのサブルーチンをアセンブルして，そのあと僕のを読み込んでアセンブルっと。

光：開始番地は6000Hだね。おっ，星が出てきた。

Ats：「流れ星」ってところですかね。ちゃんと奥行きが3段階あるでしょう。

Yo：奥の星ほど遅くなるってわけね。リスト（リスト2）を見せてくれる？

Ats：ブレイクキーを押して実行を止めて，ええと，このプログラムは大きく分けて2つの部分からできてます。77行から先で属性テーブルなんかの設定をしてます。

Yo：この#INITっていうルーチンは何？ さっき説明してもらわなかったけど。

光：HLにテーブルの先頭番地，Bにテーブルの数を入れて呼び出すと，その部分のメモリを初期化してくれるんだよね。

Yo：じゃあ，24個の星を動かしてるのね。

Ats：初期化が終わってから，今度はそれぞれのテーブルにランダムな座標とか3段階のスピードを書き込んで，初期設定は終わりです。

光：その次に飛ぶ，#STARっていうところで星を動かしてるんでしょ。

Yo：でも，これって，あとは消して動かして書いてっていうのを繰り返してるだけでしょ。

Ats：ほかに星のはみ出し処理なんかもやってますよ。42行から53行のところです。

Yo：うーん，でもねえ。

Ats：でも，なんですか？

光：工夫がないっていうんでしょ。

Yo：でしょ？ そうよねえ。

Ats：い，いや，短く収めようとしたらこんなもんですよ。

Yo：さて，こんどは光君の番ね。

光：ふっふっふっ。その点，僕の作ったのには工夫がばっちり込められてますからね。アセンブルしてJ6400，どうだ！

Ats：やられた！

光：名づけて，「宇宙気流」ってところですかね。

Yo：まあネーミングセンスは別として，ちゃんと3Dしてるわよね，キャラクタしか使っていないにしては。

Ats：星が近づくにつれて大きくなっていってる。こういうのは僕の十八番なのに。ちょっとプログラム見せてよ。

光：プログラムの流れとしては柴田君のとほとんど同じで，133行からの部分で初期設定をしてるでしょ，

Yo：40個も星を動かしてるのね。

Ats：で，設定が終わってから飛ぶ，#WASTEっていうルーチンでは何をしてるの？

光：星をばらつかせるために，文字どおり無駄に星を動かしてるんだけど，星が出てくる前に少しだけ間が開くでしょ。その原因がここなわけ。

Yo：次の#APRCHがメインルーチンね。

光：消して動かして書いてっていうのは変

リスト2

```
0000                  1        ; ##---------------##
0000                  2        ; ## SHOOTING STAR ##
0000                  3        ; ##---------------##
0000                  4        ;
0000                  5        ;    PLOGRAM  (ats)
0000                  6        ;
6000                  7        START  $6000
6000                  8        ;
6000                  9        ;  SUB RUOTINES
6000                 10        ;
6000                 11 #MOVER  EQU  $7000
6000                 12 #?BLANK EQU  $7052
6000                 13 #INIT   EQU  $7064
6000                 14 #RND    EQU  $7075
6000                 15 #WAIT   EQU  $70A6
6000                 16        ; <<  S-OS   >>
6000                 17 @GETKY  EQU  $1FD0
6000                 18 @BRKEY  EQU  $1FCD
6000                 19 @PRNT   EQU  $1FF4
6000                 20 @LOC    EQU  $201E
6000                 21        ; ##   MAIN ROUTIN   ##
6000                 22        ;
6000 CD 61 60        23        CALL   #SET
6003 CD 07 60        24        CALL   #STAR
6006 C9              25        RET
6007                 26        ;
6007                 27 #STAR
6007 CD 11 60        28        CALL   #STREAM
600A CD CD 1F        29        CALL   @BRKEY
600D C2 07 60        30        JP     NZ,#STAR
6010 C9              31        RET
6011                 32        ; ##  MANAGING STARS  ##
6011                 33        ;
6011                 34 #STREAM
6011 DD 21 00 71     35        LD     IX,$7100
6015 11 10 00        36        LD     DE,16
6018 06 18           37        LD     B,24
601A                 38 #LOOPSR          ; -- MAIN LOOI
601A 0E 20           39        LD     C," "
601C CD 53 60        40        CALL   #DRAW
601F CD 00 70        41        CALL   #MOVER
6022 DD 7E 01        42        LD     A,(IX+1)
6025 FE 28           43        CP     40      ; OUT OF
6027 DA 3E 60        44        JP     C,#STEPSR1
602A 3E 27           45        LD     A,39
602C DD 77 01        46        LD     (IX+1),A
602F CD 75 70        47        CALL   #RND
6032 E6 0F           48        AND    15
6034 67              49        LD     H,A
6035 CD 75 70        50        CALL   #RND
6038 E6 07           51        AND    7
603A 84              52        ADD    A,H
603B DD 77 02        53        LD     (IX+2),A ; SET -
603E                 54 #STEPSR1
603E DD 7E 00        55        LD     A,(IX+0)
6041 3D              56        DEC    A
6042 21 50 60        57        LD     HL,#CHTAB
6045 85              58        ADD    A,L
6046 6F              59        LD     L,A
6047 4E              60        LD     C,(HL)  ; SELECT
6048 CD 53 60        61        CALL   #DRAW
604B DD 19           62        ADD    IX,DE
604D 10 CB           63        DJNZ   #LOOPSR
604F C9              64        RET
6050 6F DF 2E        65 #CHTAB  : DM "o°."
6053                 66        ; ##  CHARACTER DRAWING  ##
6053                 67        ;        ( NULL or STAR )
6053                 68 #DRAW
6053 DD 6E 01        69        LD     L,(IX+1); -X-
6056 DD 66 02        70        LD     H,(IX+2); -Y-
6059 CD 1E 20        71        CALL   @LOC
605C 79              72        LD     A,C
605D CD F4 1F        73        CALL   @PRNT
6060 C9              74        RET
6061                 75        ;  ## SETTING STAR DATA ##
6061                 76        ;
6061                 77 #SET
6061 3E 0C           78        LD     A,12
6063 CD F4 1F        79        CALL   @PRNT   ; CLEAR SCREEN
6066 21 00 71        80        LD     HL,$7100
6069 06 18           81        LD     B,24
606B CD 64 70        82        CALL   #INIT   ; INITIALIZE
606E                 83        ;
606E DD 21 00 71     84        LD     IX,$7100; IX--START AD
6072 06 08           85        LD     B,8     ; B---TIMES
6074 0E 01           86        LD     C,1     ; C---ATTRIBUT
6076 26 10           87        LD     H,$10   ; H---DELTA -X
6078 2E 10           88        LD     L,$10   ; L---SPEED
607A                 89        ;
607A CD 90 60        90        CALL   #PUT    ; SET FRONT ST
607D 06 08           91        LD     B,8
607F 0E 02           92        LD     C,2
6081 2E 0A           93        LD     L,$0A
6083 CD 90 60        94        CALL   #PUT    ; SET MIDDLE S
6086 06 08           95        LD     B,8
6088 0E 03           96        LD     C,3
608A 2E 07           97        LD     L,$07
608C CD 90 60        98        CALL   #PUT    ; SET BACK STA
608F C9              99        RET
6090                100        ; ##  SETTING DATA AT RANDOM
6090                101        ;
6090                102 #PUT
6090 1E 10          103        LD     E,16
6092                104 #LOOPPT
6092 DD 71 00       105        LD     (IX+0),C
6095 DD 74 03       106        LD     (IX+3),H
6098 DD 75 05       107        LD     (IX+5),L
609B 3E 00          108        LD     A,0
609D DD 77 06       109        LD     (IX+6),A
60A0 CD 75 70       110        CALL   #RND
60A3 E6 1F          111        AND    31
60A5 DD 77 01       112        LD     (IX+1),A ; -X-
60A8 CD 75 70       113        CALL   #RND
60AB E6 0F          114        AND    15
60AD 57             115        LD     D,A
60AE CD 75 70       116        CALL   #RND
60B1 E6 07          117        AND    7
60B3 82             118        ADD    A,D
60B4 DD 77 02       119        LD     (IX+2),A ; -Y-
60B7 16 00          120        LD     D,0
60B9 DD 19          121        ADD    IX,DE
60BB 10 D5          122        DJNZ   #LOOPPT
60BD C9             123        RET
OBJECT CODE END 60BD
```

わらないけど，動かして書く間の処理が，柴田君のより工夫のある部分かな。

Ats：#STMNGっていうルーチンがミソになっているよね。

Yo：具体的にはどんなことをやってるの？

光：まず，星が近づくにつれて星のスピードを増してます。そうしないと，星が放射状に広がっていくだけにしか見えないんですよ。

Ats：のっぺりした星になっちゃうね。

光：そうそう。遠いものほど小さく見えるんだから，それと同時に見かけ上の移動速度も小さくしなくちゃならないんだよね。

Yo：つまり近いものほど速く動くってことね。

光：試しに84行から93行を取り除いてアセンブルすると，ほらね。星が平面にへばりついて広がっていくようにしか見えないでしょ。

Yo：なるほどね。星の移動スピードを上げることのほかには？

光：うーん，これはどちらかというと個人的な趣味の部類に属するんだけど，この星の動きをよーく見てください。

Ats：あ，星の通り道がいくつか決まってるみたいだね。

Yo：さっきの柴田君のプログラムでは，星はみんな左に動くだけだったけど，これはいろんな方向に動いてるでしょ。ということはX，Yの増分と，方向もいろいろな値を取るようになってるのよね。

Ats：たぶん動かした星が画面からはみ出したら，その星の動く方向とか速度を新しく設定し直すんだと思うんだけど……。

光：そうなんだ。で，たいていはそのXとYの増分を乱数で決めるんだろうけど，このプログラムでは違うんだな。

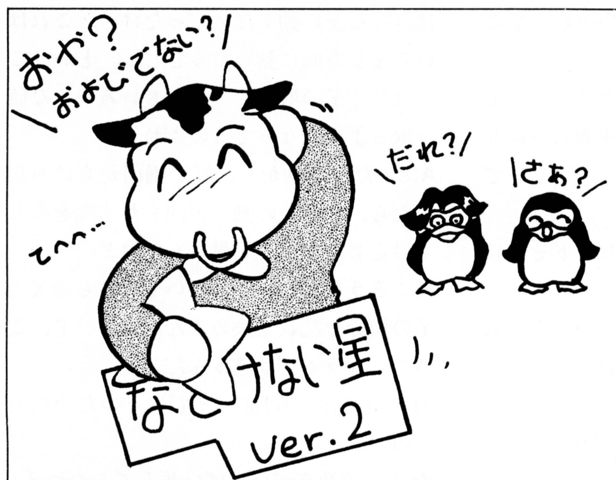
Ats：126行から130行まではそのためにあるのか。

Yo：この数字がいっぱい並んでるやつ？

光：手っ取り早くいえば，これはコサイン

リスト3

```
0000                         1       ; ##------------##
0000                         2       ; ## STAR STREAM ##
0000                         3       ; ##------------##
0000                         4       ;
0000                         5       ;    PLOGRAM  (H.M)
0000                         6       ;
6400                         7       START  $6400
6400                         8       ;
6400                         9       ;  SUB RUOTINES
6400                        10       ;
6400                        11 #MOVER  EQU  $7000
6400                        12 #?BLANK EQU  $7052
6400                        13 #INIT   EQU  $7064
6400                        14 #RND    EQU  $7075
6400                        15 #WAIT   EQU  $70A6
6400                        16       ; <<  S-OS   >>
6400                        17 @GETKY  EQU  $1FD0
6400                        18 @BRKEY  EQU  $1FCD
6400                        19 @PRNT   EQU  $1FF4
6400                        20 @LOC    EQU  $201E
6400                        21       ; ##   MAIN ROUTIN   ##
6400                        22       ;
6400 CD F9 64               23       CALL   #PRDCT
6403 CD 64 64               24       CALL   #WASTE
6406 CD 0A 64               25       CALL   #APRCH
6409 C9                     26       RET
640A                        27       ; ## APROACHING STAR ##
640A                        28       ;
640A                        29 #APRCH
640A 06 28                  30       LD     B,40
640C DD 21 00 71            31       LD     IX,$7100
6410 11 10 00               32       LD     DE,16
6413                        33 #LOOPAR        ; -- MAIN LOOP --
6413 DD 6E 01               34       LD     L,(IX+1) ; -X-
6416 DD 66 02               35       LD     H,(IX+2) ; -Y-
6419 CD 1E 20               36       CALL   @LOC
641C 3E 20                  37       LD     A," "
641E CD F4 1F               38       CALL   @PRNT   ; CLEAR STAR
6421 CD 7A 64               39       CALL   #STMNG  ; APROACHING
6424 DD 6E 01               40       LD     L,(IX+1)
6427 DD 66 02               41       LD     H,(IX+2)
642A CD 1E 20               42       CALL   @LOC
642D DD 7E 05               43       LD     A,(IX+5)
6430 CB 3F                  44       SRL    A
6432 DD 86 0A               45       ADD    A,(IX+10)
6435 DD 77 0A               46       LD     (IX+10),A
6438 CB 3F                  47       SRL    A
643A CB 3F                  48       SRL    A
643C CB 3F                  49       SRL    A
643E CB 3F                  50       SRL    A
6440 CB 3F                  51       SRL    A
6442 21 5E 64               52       LD     HL,#STTOP
6445 85                     53       ADD    A,L
6446 6F                     54       LD     L,A
6447 7E                     55       LD     A,(HL)  ; SELECT CHAR.
6448 CD F4 1F               56       CALL   @PRNT
644B DD 19                  57       ADD    IX,DE
644D 10 C4                  58       DJNZ   #LOOPAR
644F 06 FF                  59       LD     B,255
6451 CD A6 70               60       CALL   #WAIT
6454 CD A6 70               61       CALL   #WAIT   ; WASTE TIME
6457 CD CD 1F               62       CALL   @BRKEY  ; CHECK BR-KEY
645A C2 0A 64               63       JP     NZ,#APRCH
645D C9                     64       RET
645E 20 2E A1 6F 4F 4F      65 #STTOP  : DM " ..oOO"
6464                        66       ;  ###  SCATTERING STAR  ###
6464                        67       ;
6464                        68 #WASTE
6464 06 32                  69       LD     B,50
6466 11 10 00               70       LD     DE,16
6469                        71 #LOOPWT
6469 DD 21 00 71            72       LD     IX,$7100
646D 0E 28                  73       LD     C,40
646F                        74 #LOOPWT'
646F CD 7A 64               75       CALL   #STMNG
6472 DD 19                  76       ADD    IX,DE
6474 0D                     77       DEC    C
6475 C2 6F 64               78       JP     NZ,#LOOPWT'
6478 10 EF                  79       DJNZ   #LOOPWT
647A                        80       ; # MANAGING MOVEMENT OF STARS #
647A                        81       ;
647A                        82 #STMNG
647A CD 00 70               83       CALL   #MOVER  ; MOVING STAR
647D DD 34 0B               84       INC    (IX+11)
6480 DD 7E 0B               85       LD     A,(IX+11)
6483 E6 01                  86       AND    1
6485 C2 94 64               87       JP     NZ,#STEPSG
6488 DD 7E 05               88       LD     A,(IX+5)
648B 3C                     89       INC    A
648C FE 10                  90       CP     16
648E CA 94 64               91       JP     Z,#STEPSG
6491 DD 77 05               92       LD     (IX+5),A ; ADD SPEED
6494                        93 #STEPSG
6494 DD 7E 01               94       LD     A,(IX+1)
6497 FE 28                  95       CP     40
6499 D2 A2 64               96       JP     NC,#OUT
649C DD 7E 02               97       LD     A,(IX+2)
649F FE 18                  98       CP     24
64A1 D8                     99       RET    C       ; OUT OF SCREEN ?
64A2                       100 #OUT
64A2 3E 00                 101       LD     A,0
64A4 DD 77 0A              102       LD     (IX+10),A
64A7 DD 77 0B              103       LD     (IX+11),A
64AA 3E 14                 104       LD     A,20
64AC DD 77 01              105       LD     (IX+1),A ; -X-
64AF 3E 0C                 106       LD     A,12
64B1 DD 77 02              107       LD     (IX+2),A ; -Y-
64B4 CD 75 70              108       CALL   #RND
64B7 E6 0F                 109       AND    $0F
64B9 87                    110       ADD    A,A
64BA 21 D9 64              111       LD     HL,#DRTAB
64BD 85                    112       ADD    A,L
64BE 6F                    113       LD     L,A
64BF 7E                    114       LD     A,(HL)
64C0 DD 77 03              115       LD     (IX+3),A ; DELTA -X-
64C3 23                    116       INC    HL
64C4 7E                    117       LD     A,(HL)
64C5 DD 77 04              118       LD     (IX+4),A ; DELTA -Y-
64C8 CD 75 70              119       CALL   #RND
64CB E6 03                 120       AND    3
64CD DD 77 06              121       LD     (IX+6),A ; DIRECTION
64D0 CD 75 70              122       CALL   #RND
64D3 E6 03                 123       AND    $03
64D5 DD 77 05              124       LD     (IX+5),A ; SPEED
64D8 C9                    125       RET
64D9                       126 #DRTAB
64D9 00 10 02 10 03 10 05  127        DB  0:16: 2:16: 3:16: 5:15
64E0 0F
64E1 07 0F 08 0E 09 0D 0B  128        DB  7:15: 8:14: 9:13:11:12
64E8 0C
64E9 0C 0B 0D 09 0E 08 0F  129        DB 12:11:13: 9:14: 8:15: 7
64F0 07
64F1 0F 05 10 03 10 02 10  130        DB 15: 5:16: 3:16: 2:16: 0
64F8 00
64F9                       131       ; # GENERATING STARS AT RANDOM #
64F9                       132       ;
64F9                       133 #PRDCT
64F9 3E 0C                 134       LD     A,12
64FB CD F4 1F              135       CALL   @PRNT
64FE 21 00 71              136       LD     HL,$7100
6501 06 28                 137       LD     B,40
6503 CD 64 70              138       CALL   #INIT
6506 DD 21 00 71           139       LD     IX,$7100
650A 11 10 00              140       LD     DE,16
650D 06 28                 141       LD     B,40
650F                       142 #LOOPPD
650F DD 36 00 01           143       LD     (IX+0),1
6513 CD 75 70              144       CALL   #RND
6516 E6 1F                 145       AND    31
6518 DD 77 01              146       LD     (IX+1),A
651B DD 36 02 05           147       LD     (IX+2),5
651F DD 36 03 10           148       LD     (IX+3),16
6523 DD 36 05 10           149       LD     (IX+5),16
6527 DD 36 06 00           150       LD     (IX+6),0
652B DD 19                 151       ADD    IX,DE
652D 10 E0                 152       DJNZ   #LOOPPD
652F C9                    153       RET
OBJECT CODE END 652F
```



とサインのテーブルなんですよ。星が画面からはみ出したら，ここを参照してX，Yの増分を決めるんですよ。

Yo：だけどなんでわざわざそうするの？

Ats：増分を直接乱数で決めると，星の流れが変に固まったりするんだよね。

光：そうなんですよ。隙間が開いていても道筋をいくつか決めておいたほうが，星の流れがうまくばらつくんですよね。

＊　＊　＊

Yo：さて，2人のプログラムの説明が終わったところで，私の評価を発表します。

光：ツケ半分，ツケ半分！

Yo：ええと，2つのプログラムを相対的に見ると，やっぱり光君のプログラムのほうが優れてるかな。

光：でしょ，でしょ？

Ats：まあ，しょうがないな，あれじゃ。

Yo：ですが……。

光：え？　ですがなんですか？

Yo：一般のプログラムとの絶対的な評価となると，2つとも創作意欲に乏しいといわざるをえないんじゃないかしら。だってありきたりよね，どっちも。

光：ということは……。

Yo：ということで，この勝負はドロー。したがって，ツケ半分もなし！

光：そ，そりゃないですよ，ようこさん。

Ats：うーん，得したんだか損したんだかわからないや。



同盟発足

Ats：光君，ちょっと。

光：あーあ，まったくひどいよなあ，ようこさんって。

Ats：そのことでちょっと相談があるんだけど。

光：くだらない相談だったらやめてよ。もう愛想笑いする気力もないんだから。

Ats：実はね，いつかゲームに使おうと思ってたとっておきのアイデアがあるんだ。それを提供するから，光君がプログラムを作ってよ。

光：いまさら何したってもう遅いって。だいたい新しいプログラム作ったって，ようこさんが受け付けてくれなかったら，それでおしまいじゃないか。

Ats：いや，だから光君に作るのを頼むんだよ。さっきのを見てると僕なんかよりずっと早くプログラム書けるみたいだし，その速さに勝算があると思うんだけど。

光：速さには自身があるけどね。だったらやってみてもいいけど。

Ats：よし，そうこなくっちゃ。ようこさんようこさん！

Yo：おだてたって，さっきの評価は取り消しませんからね。

光：そんなせこいことしませんよ。

Ats：あのですね，もう一度だけチャンスをもらいたいんですけど。それも今度は僕と光君の合作ということで。

Yo：ええ，だめよ。だってまた1時間かそこらの時間がかかるんでしょ。マスターだって帰ってくるし，ほかのお客さんにも迷惑よ。

Ats：いいえ，こんどは30分で作ってみせます。

Yo：ほほー，いったわね。いくら2人がかりでも30分は無理じゃないの？

光：じゃあ，もし30分でできなかったら，このことはなかったことにしてもらっていいですよ。

Yo：まあ，そういう条件ならいいかな。でも30分を少しでも超えたら失格ね。

Ats：やった！　そうと決まればさっさと始めよう。

光：でも，30分はふっかけすぎじゃないのかなあ。

Ats：大丈夫だって。イメージもかなり固まってるし，あとはそれをコードに落とすだけなんだから。



そして，28分後

光：確かめもせずに一気にプログラム書いてきたけど，これでバグが出なかったらほとんど奇跡だよ。

Ats：とにかくアセンブルして走らせてみよう。6800_Hからだったよね。

光：スペースキーを押すと花火が上がるはずだけど，あーっ！　キーリピートがかかってる！

Ats：ええと，最後に押されたコードといま押してるキーのコードをCPして，同じだったらキー入力に戻る！

マスター(以下M)：ただいま。おや，みなさんおそろいで。

Yo：あっ，マ，マスター。予定より早いんじゃないの，帰ってくるのが。

光：やったー！　できたー！（リスト4）

M：角の花屋があるでしょ，あそこの店主がおしゃべりでね。今日はその店主が来なかったんで，早く終わったんですよ。

Ats：どっかん，どっかん！

M：光君と柴田君は何やってんですか？　お，なんか作ったんですね。

光：見てくださいよ，マスターにようこさん。スペースキーを押すとね。

Yo：あら！　面白いわね。

光：ひとつだけじゃないんですよ。

Ats：こんなふうに何回もスペースキーを押したって。

M：おおっ！　けっこう頑張るじゃないですか。花火の雰囲気がよく出てますよ，季節外れだけど。どうやってるんですか？

光：むかしBEMSっていうパッケージがありましたよね。あんな感じのサブルーチンを使って動かしてるんですよ。

Yo：これだといくつぐらいのキャラを動かしてるの？

Ats：最高で130個だったと思います。花火4，5発なら持ちこたえるんじゃないですか。

Yo：ということは，何個動かすかは決まってないんだ。いままでのだとちゃんとキャラの数が決まってて，メインループでもその数だけ繰り返してたでしょう。

光：基本的な考え方は前の2つのプログラムと同じなんですけど，キャラクタの管理

の方法が１段階高級なんですよ。いままでのは30個なら30個，全部動いてるっていう仕組みだったけど，このプログラムの場合はですね。

Ats：それぞれのテーブルに，そのキャラが生きてるか死んでるかがわかるようなコードを持たせるんですよ。テーブルのいちばん最初，０番目の値が０だったら死んでいて，それ以外だったら生きているっていう約束事を，最初に決めておくんです。

M：なるほど。テーブルの０番目の値を見てみて，ゼロだったら何もせずに飛ばせばいいんだからな。

光：それに新しいキャラを作りたいとき，たとえばここでは，花火が破裂して火の粉が飛び出すときなんかは，先頭が０のテーブルを探して，そこに値をぶっこむだけでいいんですよ。

Ats：そのためのサブルーチンが#?BLANKっていうやつなんです。あ，それと，使い終わったキャラのテーブルはちゃんと先頭を０にしておかないと，ゴミがたまってすぐに動かなくなるから注意が必要ですね。

光：電気消して暗くしてみませんか。

Yo：花火の色も白より黄色のほうが感じが出るわね。

M：ああ，なんかいいですね。花火の大きさもまちまちで，奥行きがありますね。

Ats：花火の飛び散る方向はみんな同じなんですよ。でも飛び散る速度を変えてあるから，輪が大きくなったり小さくなったりするんですよね。

Yo：さっきいってた，増分とスピードが別々に設定できると便利だっていうのは，そういうことだったのね。

光：ところで，柴田君はこの花火を使ってどんなゲーム作ろうと思ってたの？

M：それは私も大いに興味がありますね。

Ats：爆発がやたらに派手なシューティングゲーム……。

Yo：聞いた光君が馬鹿だったわね。

Ats：いわなきゃよかった。

＊　＊　＊

M：さて，そろそろほかのお客さんも来る頃ですし，電気つけますよ。

Ats：ようこさんの判定はどうなったんですか。ちゃんと30分以内に作ったでしょ。

光：そうですよ。スポンサーのマスターがいることですし，ここでガツーンと。

Yo：あ，あの，そのことなんだけどね。

M：判定？ 私が何のスポンサーなんですか？

光：やだなー，マスターとぼけちゃって。僕たちにプログラミングの勝負をさせて，よくできたほうのツケを半分にするって。

Ats：審判はようこさんに任せるって。

M：ああ，同じサブルーチンを使ったプログラムを，２人に作ってもらったらっていうのは確かに私のアイデアですよ。商店会に出かけている間に２人が揃ったら，頼んでみてくれともいいました。でもツケ半額とか勝負だとか，そんなの知らないなあ。

光：え，柴田君，ツケ半分ってマスターから直接聞いたんじゃないの？

Ats：僕が来たときはもうマスター出かけてたし，僕はようこさんから……。

光：ということは，ようこさん！

Yo：あ，あのね，わたしマスターから話聞いたとき，すごく面白そうだなって思ったの，だけどタダだったら面倒臭がってやらないだろうし。

Ats：当たり前ですよ！

Yo：それに，適当にごまかしちゃえばいいかなあ，なんて思ったりしてね。もう，マスターったら気が利かないんだから！

M：私のせいじゃないのに……。

光：ということは，最初からドローにするつもりだったんだな。

Yo：悪気があってやったんじゃないのよ。

Ats：ううっ，許せん！ よし光君，フォーメーションAだっ！

光：オウ！

Yo：オウって光君，なにもカタカナで……。ああっ，何すんの，えっちっ！

光：はっはっはっ。これで動けまい。

Yo：ちょっと，わたしの靴脱がせて，どうしようっていうのよ。まさかそれをわたしにかがせるんじゃ……。

Ats：ふっ，僕たちのことを甘く見てますね。靴の臭いをかがせるんじゃない！

Yo：ということは……。

Ats：僕がかぐんだっ！

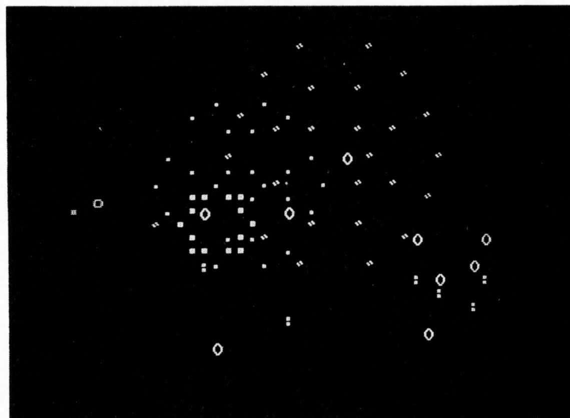
光：わっはっはっ，どうだ，クサハズカシイだろう。

Yo：ひぃー。

Ats：わっはっはっ。

M：結局，今月もこうなるのか。

―つづく―



FIREWORK

リスト4

```
0000                  ; ##--------------##
0000                  ; ##   FIREWORK   ##
0000                  ; ##--------------##
0000                  ;
0000                  ;      PLOGRAM  (a&H)
0000                  ;
6800                 START  $6800
6800                  ;
6800                  ;   SUB RUOTINES
6800                  ;
6800           #MOVER  EQU  $7000
6800           #?BLANK EQU  $7052
6800           #INIT   EQU  $7064
6800           #RND    EQU  $7075
6800           #WAIT   EQU  $70A6
6800                  ; <<  S-OS   >>
6800           @GETKY  EQU  $1FD0
6800           @BRKEY  EQU  $1FCD
6800           @PRNT   EQU  $1FF4
6800           @LOC    EQU  $201E
6800                  ; ##   MAIN ROUTIN   ##
6800                  ;
6800 3E 0C             LD     A,12
6802 CD F4 1F          CALL   @PRNT
6805 21 00 71          LD     HL,$7100
6808 06 82             LD     B,130
680A CD 64 70          CALL   #INIT   ; INITIALIZE
680D CD 11 68          CALL   #WORK
6810 C9                RET
6811                   ;
6811           #WORK
6811 32 32 68          LD     (#LAST),A
6814 CD 33 68          CALL   #FIRE
6817 CD D0 1F          CALL   @GETKY
681A FE 1B             CP     $1B     ; CHECK BR-KRY
681C C8                RET    Z
681D FE 20             CP     " "
681F C2 11 68          JP     NZ,#WORK
6822 4F                LD     C,A
6823 3A 32 68          LD     A,(#LAST)
```

```
6826 B9                 CP      C        ; NOT TO REPEAT
6827 CA 11 68           JP      Z,#WORK
682A CD F4 68           CALL    #PREP    ; SET FIREWORK
682D 3E 20              LD      A," "
682F C3 11 68           JP      #WORK
6832 00          #LAST   : DB 0  ; KEY BUFF.
6833                    ; ##   MANAGING FIREWORK   ##
6833                    ;
6833             #FIRE
6833 DD 21 00 71        LD      IX,$7100
6837 11 10 00           LD      DE,16
683A 0E 82              LD      C,130
683C             #LOOPFI         ; -- MAIN LOOP --
683C DD 7E 00           LD      A,(IX+0)
683F FE 00              CP      0
6841 CA E1 68           JP      Z,#SKIP1 ; CHECK ATTR.
6844 47                 LD      B,A
6845 DD 6E 01           LD      L,(IX+1)
6848 DD 66 02           LD      H,(IX+2) ; -Y-
684B CD 1E 20           CALL    @LOC
684E 3E 20              LD      A," "
6850 CD F4 1F           CALL    @PRNT    ; CLEAR CHAR.
6853 CD 00 70           CALL    #MOVER   ; MOVING
6856 78                 LD      A,B
6857 FE 03              CP      3
6859 CA A7 68           JP      Z,#SPARK ; SPARK
685C DD 34 0A           INC     (IX+10)
685F FE 02              CP      2
6861 CA 87 68           JP      Z,#SMOKE ; SMOKE
6864                    ; MANAGING FIRE BALL
6864                    ;
6864 DD 7E 0A           LD      A,(IX+10)
6867 FE 1E              CP      30
6869 DA 76 68           JP      C,#STEPFI1
686C CD 54 69           CALL    #BURST   ; EXPLOSE
686F DD 36 00 00        LD      (IX+0),0
6873 C3 E6 68           JP      #SKIP2
6876             #STEPFI1        ; FIRE GOIN' UP
6876 DD 6E 01           LD      L,(IX+1)
6879 DD 66 02           LD      H,(IX+2)
687C CD 1E 20           CALL    @LOC
687F 3E 4F              LD      A,"O"
6881 CD F4 1F           CALL    @PRNT
6884 C3 E6 68           JP      #SKIP2
6887                    ; MANAGING BALL'S TAIL
6887                    ;
6887             #SMOKE
6887 DD 7E 0A           LD      A,(IX+10)
688A FE 14              CP      20
688C DA 96 68           JP      C,#STEPFI2
688F DD 36 00 00        LD      (IX+0),0
6893 C3 E6 68           JP      #SKIP2
6896             #STEPFI2
6896 DD 6E 01           LD      L,(IX+1)
6899 DD 66 02           LD      H,(IX+2)
689C CD 1E 20           CALL    @LOC
689F 3E 3A              LD      A,":"
68A1 CD F4 1F           CALL    @PRNT
68A4 C3 E6 68           JP      #SKIP2
68A7                    ; MANAGING SPARK
68A7                    ;
68A7             #SPARK
68A7 3E 27              LD      A,39
68A9 DD BE 01           CP      (IX+1)
68AC DA DA 68           JP      C,#STEPFI3
68AF 3E 17              LD      A,23
68B1 DD BE 02           CP      (IX+2)
68B4 DA DA 68           JP      C,#STEPFI3 ; OUT OF SCR.?
68B7 DD 34 0A           INC     (IX+10)
68BA DD 7E 0A           LD      A,(IX+10)
68BD FE 0E              CP      14
68BF D2 DA 68           JP      NC,#STEPFI3
68C2 21 ED 68           LD      HL,#SPTOP
68C5 CB 3F              SRL     A
68C7 85                 ADD     A,L
68C8 6F                 LD      L,A
68C9 46                 LD      B,(HL)   ; SELECT CHAR.
68CA DD 6E 01           LD      L,(IX+1) ; -X-
68CD DD 66 02           LD      H,(IX+2) ; -Y-
68D0 CD 1E 20           CALL    @LOC
68D3 78                 LD      A,B
68D4 CD F4 1F           CALL    @PRNT
68D7 C3 E6 68           JP      #SKIP2
68DA             #STEPFI3        ; OUT OF SCREEN
68DA DD 36 00 00        LD      (IX+0),0
68DE C3 E6 68           JP      #SKIP2
68E1             #SKIP1
68E1 06 28              LD      B,40
68E3 CD A6 70           CALL    #WAIT    ; WEAST TIME
68E6             #SKIP2
68E6 DD 19              ADD     IX,DE
68E8 0D                 DEC     C
68E9 C2 3C 68           JP      NZ,#LOOPFI
68EC C9                 RET
68ED 20 20 6F DF A5 DE A4  #SPTOP  : DM "  oﾟ･ﾞ､"
68F4                    ; ##  PREPARING FOR SETTING  ##
68F4                    ;
68F4             #PREP
68F4 C5                 PUSH    BC
68F5 D5                 PUSH    DE
68F6 E5                 PUSH    HL
68F7 06 82              LD      B,130
68F9 21 00 71           LD      HL,$7100
68FC CD 52 70           CALL    #?BLANK ; FINDING BLANK
68FF DA 45 69           JP      C,#NOPRP ; NO BLANK
6902 EB                 EX      DE,HL
6903 21 49 69           LD      HL,#DATA
6906 01 0B 00           LD      BC,11
6909 ED B0              LDIR             ; TRANSFER DATA
690B EB                 EX      DE,HL
690C B7                 OR      A
690D 01 0A 00           LD      BC,10
6910 ED 42              SBC     HL,BC
6912 CD 75 70           CALL    #RND
6915 E                  AND     31
6917 C                  ADD     A,4
6919 77                 LD      (HL),A   ; -X-
691A 01                 LD      BC,4
691D 09                 ADD     HL,BC
691E CD 75 70           CALL    #RND
```

```
6921 E6 07              AND     7
6923 C6 09              ADD     A,9
6925 77                 LD      (HL),A  ; SPEED
6926 B7                 OR      A
6927 ED 42              SBC     HL,BC
6929 EB                 EX      DE,HL
692A 21 00 71           LD      HL,$7100
692D 06 82              LD      B,130
692F CD 52 70           CALL    #?BLANK ; -- FOR SMOKE --
6932 DA 45 69           JP      C,#NOPRP
6935 36 02              LD      (HL),2
6937 23                 INC     HL
6938 EB                 EX      DE,HL
6939 01 0A 00           LD      BC,10
693C ED B0              LDIR
693E 01 07 00           LD      BC,7
6941 B7                 OR      A
6942 ED 42              SBC     HL,BC
6944 34                 INC     (HL)
6945             #NOPRP
6945 C1                 POP     BC
6946 D1                 POP     DE
6947 E1                 POP     HL
6948 C9                 RET
6949 01 00 16 00 08 00   #DATA : DB  1: 0:22: 0: 8: 0
694F 00 00 00 00 00             DB  0: 0: 0: 0: 0
6954                    ;  ##  EXPLOSING FIREWORK  ##
6954                    ;
6954             #BURST
6954 C5                 PUSH    BC
6955 D5                 PUSH    DE
6956 E5                 PUSH    HL
6957 7D                 LD      A,L
6958 32 E2 69           LD      (#STOCK+1),A ; -X-
695B 7C                 LD      A,H
695C 32 E3 69           LD      (#STOCK+2),A ; -Y-
695F 06 04              LD      B,4
6961             #LOOPBR1        ; FOR OUTSIDE
6961 21 EC 69           LD      HL,#DRTOP
6964 0E 04              LD      C,4
6966             #LOOPBR2
6966 7E                 LD      A,(HL)
6967 32 E4 69           LD      (#STOCK+3),A ; DELTA -X-
696A 23                 INC     HL
696B 7E                 LD      A,(HL)
696C 32 E5 69           LD      (#STOCK+4),A ; DELTA -Y-
696F 23                 INC     HL
6970 DD 7E 05           LD      A,(IX+5)
6973 32 E6 69           LD      (#STOCK+5),A ; SPEED
6976 78                 LD      A,B
6977 E6 03              AND     3
6979 32 E7 69           LD      (#STOCK+6),A ; DIRECTION
697C CD C6 69           CALL    #LOAD    ; TRANSFER DATA
697F 0D                 DEC     C
6980 C2 66 69           JP      NZ,#LOOPBR2
6983 10 DC              DJNZ    #LOOPBR1
6985 16 02              LD      D,2
6987 DD 7E 05           LD      A,(IX+5)
698A CB 3F              SRL     A
698C C6 03              ADD     A,3
698E 5F                 LD      E,A
698F             #LOOPBR3        ; FOR INSIDE
698F 06 04              LD      B,4
6991             #LOOPBR4
6991 21 EC 69           LD      HL,#DRTOP
6994 0E 02              LD      C,2
6996 CB 42              BIT     0,D
6998 C2 9D 69           JP      NZ,#LOOPBR5
699B 23                 INC     HL
699C 23                 INC     HL
699D             #LOOPBR5
699D 7E                 LD      A,(HL)
699E 32 E4 69           LD      (#STOCK+3),A
69A1 23                 INC     HL
69A2 7E                 LD      A,(HL)
69A3 32 E5 69           LD      (#STOCK+4),A
69A6 23                 INC     HL
69A7 23                 INC     HL
69A8 23                 INC     HL
69A9 7B                 LD      A,E
69AA 32 E6 69           LD      (#STOCK+5),A
69AD 78                 LD      A,B
69AE E6 03              AND     3
69B0 32 E7 69           LD      (#STOCK+6),A
69B3 CD C6 69           CALL    #LOAD
69B6 0D                 DEC     C
69B7 C2 9D 69           JP      NZ,#LOOPBR5
69BA 10 D5              DJNZ    #LOOPBR4
69BC CB 3B              SRL     E
69BE 15                 DEC     D
69BF C2 8F 69           JP      NZ,#LOOPBR3
69C2 C1                 POP     BC
69C3 D1                 POP     DE
69C4 E1                 POP     HL
69C5 C9                 RET
69C6                    ;  ##  TRANCEFERING DATA  ##
69C6                    ;
69C6             #LOAD
69C6 E5                 PUSH    HL
69C7 D5                 PUSH    DE
69C8 C5                 PUSH    BC
69C9 06 82              LD      B,130
69CB 21 00 71           LD      HL,$7100
69CE CD 52 70           CALL    #?BLANK
69D1 DA DD 69           JP      C,#NOBLK ; NO BLANK
69D4 EB                 EX      DE,HL
69D5 21 E1 69           LD      HL,#STOCK
69D8 01 0B 00           LD      BC,11
69DB ED B0              LDIR
69DD             #NOBLK
69DD C1                 POP     BC
69DE D1                 POP     DE
69DF E1                 POP     HL
69E0 C9                 RET
69E1 03          #STOCK :DB 3
69E2 00 00 00 00 00 00 00        DS 10
69E9 00 00 00
69EC 10 00 0F 06 0B 0C 06  #DRTOP :DB 16: 0:15: 6:11:12: 6:15
69F3 0F
OBJECT CODE END 69F3
```

THE SENTINEL

〈対応機種一覧〉●MZ-80K/C/700/1500●MZ-80B/2000●MZ-2500/2861●X1●X1turbo/Z●PC-8001/8801/88●SMC-777/C●PASOPIA/5●PASOPIA 7●FM-7/77/AV●PC-286/386/9801/'98●X68000
掲載されたプログラムの利用には各機種用のS-OS"SWORD"システムが必要です。

第114部 Small-C用SLANGコンパチ関数

●Small-C用SLANGコンパチ関数

Human68kやMS-DOSなどでは，ESC文字に続く一連の文字列を表示することで，カーソル位置や文字の色などを指定することができます。この機能はエスケープシーケンスと呼ばれており，これを利用すればC言語の標準ライブラリに用意されているprintfなどの文字表示関数を使って画面を制御することができます。

エスケープシーケンスのないS-OSでは，コントロールコードを使っても画面を消去することしかできません。もちろんこれでは，ゲームを作ることなど夢のまた夢です。S-OSが持っているカーソル位置指定やWIDTH変更用などのルーチンをサポートするライブラリがあればどんなに便利だろう……という話は，このSENTINELでも何度かしましたね。ついに今月，この要求に沿ったライブラリが読者の手によって提供されることとなりました。

今回のライブラリによって用意されたのは，純粋にS-OSの共通ルーチンを利用するものではなく，S-OSの標準整数型コンパイラともいえるSLANGの関数に準拠したものです。前述の画面制御に加え，S-OSの#GETKY，#FLGET，#INKEYに対応するキー入力，10進数のINPUTなども用意されています。

Small-C用のライブラリには，数学関係の関数が一切入っていません(整数型なので，当然といえば当然なのですが)。しかし，ゲームを作るとなると，乱数ルーチンくらいは欲しいところです。これも今回のライブラリに収録されています。そのほかビット操作関係の関数も用意され，かなり便利に使えるのではないでしょうか。

●サンプルプログラムもぐらたたき

このライブラリは，ゲーム作りを意識して発表されたものといえるでしょう。もちろん，サンプルゲームつき。題材はなつかしのもぐらたたきです。

追加関数は，それぞれ別個にWZDでアセンブルしてリロケータブルオブジェクトにし，WLBでライブラリにまとめて使います。全部を入力するのが面倒だという方は，必要なものだけを入力してリロケータブルオブジェクトを作り，とりあえずミニマムのライブラリを作ってしまってはいかがでしょうか。以後必要な関数が出てくるたびにこのライブラリに追加していくという方法なら，それほど負担にならずにライブラリを仕上げることができるでしょう。

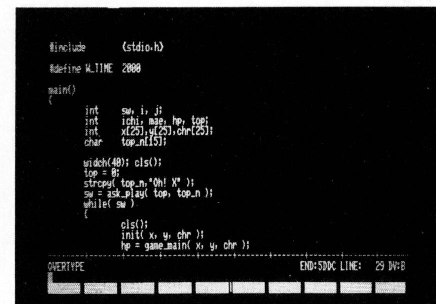
Small-CはミニマムなCコンパイラですから，皆さんがちょっと凝ったプログラムを作成する場合には，アセンブラの力を借りなければならない場合も出てくると思います。そんなときに用意した関数を投稿していただけると，Small-Cをとりまく環境はますます充実していくことになります。なければ，作る。そしてそれをみんなで活用する。この精神です。皆さんの投稿お待ちしています。

●S-OSの系譜(27)

1988年2月号は，ちょっとひと息ついてゲームの登場です。ELFESと名づけられたこのゲームは，S-OSのキャラクタ画面を使って3Dシューティングゲームを作ってしまったという力作です。星が流れビームが飛びかうスペースシューティングゲームながら超高速！ 起動時には32段階ある速度調整が最低速になっているというのですから驚きです。X1やMZ-2000といった8ドットキャラクタの画面では，32段階ある速度調整を下手に動かすと速すぎてゲームにならないほどです。これが，専用ルーチンを一切使わず，遅いと評判のS-OSの共通ルーチンだけで実現されているというのです。アルゴリズムという言葉の意味を，改めて考えさせられる作品です。

続く3月号では，構造型コンパイラ言語SLANGが発表されました。S-OS初のオリジナルコンパイラ言語の登場です。この言語が与えた影響は，いまさらここで述べるまでもないでしょう。今月のSmall-C用のライブラリでも参照されているように，現在のS-OSシステムの中核となる整数型コンパイラ言語です。大きなプログラムを開発するときに便利なファイル取り込み(インクルード)機能やチェイン機能を揃える一方で，関数にパラメータを渡すときに，パラメータの数が3個以下の場合にはレジスタを使用するといった高速化も配慮されています。今日の支持の裏には，こういったユーザー本位の配慮が溢れていることも忘れることはできません。

「やはり生成されるオブジェクトコードや実行速度は，アセンブラで記述されたものと見まがうばかりでなければなりません」という作者の大貫氏の心意気がここにあります。

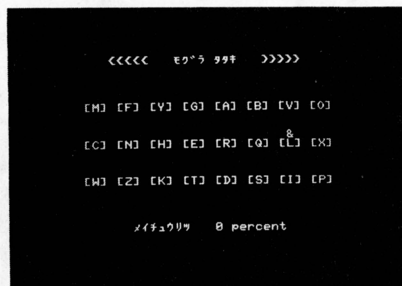


全機種共通
S-OS“SWORD”要

Small-C
SLANGコンパチ関数

Itou Naoya
伊藤 直也

Small-CにSLANGコンパチ関数のライブラリが登場です。これによってSmall-CからS-OSのおいしいファンクションコールを簡単に利用することができます。好みにあわせて活用しましょう。



最初に，私がこれらの関数を作ろうと思ったのは，Small-CがSLANGに比べて少し使いづらいと感じたからです。たとえば画面のある場所に文字を表示したくても，そうするための関数がありません。これでは私が得意とする小細工の利いたプログラムを作成できないではないか。そんなことがきっかけで，このライブラリを作ることを決意しました。

ライブラリの作成

この関数を使うには2つの方法が考えられます。まず，あなたがプログラミング中にこのライブラリの関数が必要になったとき，関数をそのままプログラムの中に組み込んでしまう方法。もうひとつは，ここに掲載されている関数をすべて入力しておいて，ライブラリファイルにしてしまう方法があるでしょう。

ほかにもちょっとずつライブラリファイルに関数を足していく，というのも考えられます。しかし，同じ関数を2度入力してしまったり，リズムにのってプログラミングしていたのに(そんなヤツいるか？)，ライブラリが足りなかったばっかりに，作業を一時中断しなければならなくなるような場合におちいったりすることもあるでしょう。

それでも，細切れに入力しようとする人は，関数rnd(n) よりも前に関数rnd_sub( )をリンクしないでください。これはリンカWLKの仕様によるもので，未定義ラベルの検索を一方向にしか行わないためです。関数rnd( ) よりも前に関数rnd_sub( ) をリンクしてしまうと，関数rnd( ) で未定義ラベルとして，rnd_subをリクエストしても，もうrnd_subは後ろにないわけでリンカは未定義エラーとしてしまいます。

ここでは，一度にすべての関数を入力してしまう方を対象にライブラリファイルの作成を説明します。

まず，リスト1にある関数をひととおり入力してください。アセンブラで書かれた関数はアセンブルを，C言語で書かれた関数はコンパイル&アセンブルをします。この作業は，もういやというほどやっているはずなので省略します（マスターしていない人はバックナンバーを調べましょう）。

すべての関数をリロケータブルファイルに落とし終わったら，いよいよライブラリファイルにまとめます。一度，この作業は皆さんがやっているはずなのですが，もう忘れしまった方や，Small-Cのシステム一式が編集部のプレゼントで当たった方もいるかと思いますので書いておきます。

同じディスクにリロケータブルファイルに落とした関数一式とライブラリアンWLBを入れておきます。そうしたら以下のコマンドを実行するだけです。

```
#LWLB
#J3000
*cls,locate,widch,screen,inkey,getl
*getlin,linput,input,prmode,rnd
*rnd_sub,beep,sgn,sex,bit,set,reset
*peek,peekw,poke,pokew
*slnlib/r
```

以上で今回掲載した関数一式をまとめたライブラリファイルslnlib.LIBが作成されます。これらの関数を使用するときは，プログラムのリンク時にライブラリファイルclib.LIBをリンクする前にslnlib.LIBをリンクさせます。

たとえば，main.Cというファイルに今回の関数を使用しているプログラムがあるとすると，これをリンクするときは，

```
#LWLK
#J3000
*/P:3000,main,slnlib/s,clib/s,main/
n:p
```

とします。

さらに，今回のslnlib.LIBをclib.LIBに入れてしまうこともできます。

```
#LWLB
#J3000
*slnlib,clib,clib1/r
```

で，今回の関数と標準的な関数をすべて含んだライブラリファイルclib1.LIBが作成されます。

サンプルプログラム

今回の関数を使った，サンプルプログラムを紹介します。今回の拡張はどちらかというと，キーボード，画面周りの拡張なので，ここらへんを使ったサンプルとなるとやはりゲームでしょう。今回作成したのは，TK-80BSの昔からあるゲーム界の三葉虫

「モグラ叩き」です（リスト2）。

ルールはいたって簡単。モグラの現れるところに，ハンマーを打ち込みモグラを退治するだけです。ハンマーが見事モグラを叩くと命中率が上がり，外れると命中率が下がります。モグラが30匹出現すると，ゲームオーバーになり，このときの命中率を競います。タイマはX1に合わせてあるので，初代PC-8801などをお持ちの方は21行目の，

#define W_TIME 20000

の値を調節してみてください。

*　　*　　*

とりあえず関数も整い始めましたので，今度は何かゲームでも作ってみようかと思っています。

私はCもアセンブラも苦手なほうなので，ここはこうしたほうがいい，というような意見があればどんどん改良してよりいいものを作っていきましょう。ぜひ，みなさんの知恵をお借りしたいものです。

表1　SLANGコンパチ関数仕様書

・cls（　）
画面を消去する。
・locate (x,y)
x，yの位置にカーソルを移動する。
・widch (n)
画面モードを切り替えます。nが40以下で40桁に，40を超えると80桁になります。
・screen (x,y)
画面上のx，y座標の位置にあるキャラクタコードを返り値とします。
・inkey (n)
入力されたキーコードを返り値とします。
n＝0……S-OSの#GETKY
n＝1……S-OSの#FLGET
その他……S-OSの#INKEY
・getl(格納アドレス)
キーボードから格納アドレスに1行入力して入力された行の長さを返り値とします。BREAKキーが押されたら−1を返します。行の最後は0となっています。
・getlin(格納アドレス，長さ)
指定された長さで1行入力を行います。ほかはgetl（　）と同じです。
・linput(格納アドレス，長さ)
コールされた時点のカーソル以降にある画面上の文字列を格納アドレスに取り込みます。
・input（　）
キーボードから入力された10進数を返り値とします。先頭に“$”をつけると16進数とみなします。
※正常な入力が行われたかのチェックはしません。
・prmode (n)
nが0のときは画面のみの出力，nが0以外のときには画面・プリンタに出力します。
・rnd (n)
n＞0……0〜n−1の乱数を返します
n＝0……0を返します
n＜0……0〜n＋1の乱数を返します
※nは−32768〜32767の範囲です。
・rnd_sub(　)
1〜65535の乱数を返り値とします。
・beep（　）
BEEP音を鳴らします。
・sgn (n)
nの符号を返り値とします。
n＞0……1を返します
n＝0……0を返します
n＜0……−1を返します
・sex (n)
nを符号付1バイトの値とみなし，符号付2バイトの値を返り値とします。
・bit(値，n)
値の第nビットを調べて0か，1を返します。nは0〜15の範囲です。
・set(値，n)
値の第nビットを1にします。
・reset(値，n)
値の第nビットを0にします。
・peek(アドレス)
アドレスから1バイトの値を読み出して，それを返り値とします。
・peekw(アドレス)
アドレスから2バイトの値を読み出して，それを返り値とします。
・poke(アドレス，値)
アドレスに1バイトの値を書き込みます。
・pokew(アドレス，値)
アドレスに2バイトの値を書き込みます。

今月のバグだし

大野城市の浜地啓さんより
○拡張した場合，コマンドラインにおいて“：”のあとにスペースがなくてはならない。

そういう仕様なのですが，記事の中で一部空白が載っていませんでした。Small-C活用講座の記事において，

CC:ファイル名

とあるのは，

CC:　ファイル名

だと思ってください。

横浜市の高山慎一さんより
○8月号に掲載もれのリストがあった。

以下のリストです。ライブラリディスク4に入れておいてください。

```
                delete.ASM
    delete::
            EXT         unlink
            END
```

浜松市の伊藤直也さん，富士見市の樫正一郎さんほか，大勢の皆さんより
○10月号の変更を行うと出力ファイルの名前がおかしくなる。

・ソースリストでの変更方法

ファイルCC11.Cの中の関数openfile（　）をリスト3のように変更する。

・オブジェクトファイルに直接変更を加える

ただし，この方法だとファイル名の長さが15文字を超えた場合の動作は保証できません。絶対に15文字以上のファイル名を与えないでください。訂正はリスト4のダンプリストを入力してから，以下の変更を行ってください。

3CB6　CE 3F→8F 40
（もしくは50 3D→8F 40）

以下は，富士見市の樫正一郎さんより
○ungetc（　）関数が標準入力に対応していなかった(二次的に関数scanf（　）において，最初の1文字が無視されてしまう)。

ファイルungetc.Cをリスト5のように変更してから，rdrtl.asm内のサブルーチンgetcharを以下のように変更してください。

```
;
;       getchar ( )
;
  getchar::
            LD      A,(__con)
            AND     A
            JR      NZ,getch1
            CALL    _FLGET
            CALL    _PRINT
  getch1:   LD      L,A
            LD      H,0
            CP      CTRL_Z
            RET     NZ
            LD      HL,-1
            RET
```

また，同じファイルrdrtl.asmのラベルSETIO:の直後に以下の2行を，

```
            XOR     A
            LD      (__con),A
```

追加して，さらにEND文の直前に，

```
    __con::  DS      1
```

を追加してください。

○Eドライブをサポートしていないので，RAMディスクが使えない。

rdrtl.asm内のサブルーチンdevchkを以下のように変更してください。

```
    DEVCHK:CP       'A'
            JR      C,DEVCHK1
            CP      'E'+1
            CCF
            RET     NC
   DEVCHK1: LD      A,3
            RET
```

情報を提供してくださった皆様ありがとうございました。　（石上達也）

リスト1

```
;====================================================
;       cls()              by N.ito  '91/9/21
;====================================================

_PRINT  EQU    1FF4H

cls::
        LD     A,0CH
        CALL   _PRINT
        RET


```

```
;====================================================
;       locate(x,y)        by N.ito  '91/9/21
;====================================================

_LOC    EQU    201EH

locate::
        INC    SP         ; Skip over return address
        POP    BC         ; Y
        LD     H,B
        POP    DE         ; X
        LD     L,D
        PUSH   DE
        PUSH   BC
        DEC    SP
        CALL   _LOC
        RET

```

```
;====================================================
;       widch(n)           by N.ito  '91/9/22
;====================================================

_WIDCH  EQU    2030H

widch::
        POP    BC         ; Skip over return address
        POP    DE         ; 40 or 80
        LD     A,E
        PUSH   DE
        PUSH   BC
        CALL   _WIDCH
        RET

```

```
;====================================================
;       screen(x,y)        by N.ito  '91/9/22
;====================================================

_SCRN   EQU    201BH

screen::
        INC    SP         ; Skip over return address
        POP    BC         ; Y
        LD     H,B
        POP    DE         ; X
        LD     L,D
        PUSH   DE
        PUSH   BC
        DEC    SP
        CALL   _SCRN
        LD     L,A
        LD     H,0
        RET

```

```
;====================================================
;       inkey(n)           by N.ito  '91/9/22
;====================================================

_GETKEY EQU    1FD0H
_FLGET  EQU    2021H
_INKEY  EQU    1FCAH

inkey::
        POP    BC         ; Skip over return address
        POP    DE         ; mode
        LD     A,E
        PUSH   DE
        PUSH   BC
        OR     A              ; CP 0
        JR     NZ,SKIP1   ; n >= 1
        CALL   _GETKEY    ; n =  0
        JR     TERM
SKIP1:
        CP     1
        JR     NZ,SKIP2   ; n >= 2
        CALL   _FLGET     ; n = 1
        JR     TERM
SKIP2:
        CALL   _INKEY         ; n >= 2
TERM:
        LD     L,A
        LD     H,0
        RET

```

```
;====================================================
;       getl( d, n )       by N.ito  '91/9/23
;====================================================

_GETL   EQU    1FD3H
_KBFAD  EQU    1F76H

getl::
        LD     DE,(_KBFAD)
        CALL   _GETL
        INC    SP
        INC    SP
        POP    BC                          ; arr top
        PUSH   BC
        DEC    SP
        DEC    SP
        PUSH   BC                          ; save arr top

        LD     HL,0
GL_LOOP:
        LD     A,(DE)                      ; key buf --/
        LD     (BC),A                      ;          /--> arr
        OR     A
        JR     Z,GL_SKIP
        INC    DE
        INC    BC
        INC    L
        JR     GL_LOOP
GL_SKIP:
        POP    BC
        LD     A,(BC)
        CP     1BH                         ; top = 1B ?
        RET    NZ
        LD     A,-1
        LD     (BC),A                      ; 1Bh -> -1
        LD     HL,-1                       ; cnt
        RET

```

```
;====================================================
;       getlin( d, n )     by N.ito  '91/9/23
;====================================================

_GETL   EQU    1FD3H
_KBFAD  EQU    1F76H

getlin::
        LD     DE,(_KBFAD)
        CALL   _GETL
        INC    SP
        INC    SP
        POP    HL                          ; max
        POP    BC                          ; arr top
        PUSH   BC
        PUSH   HL
        DEC    SP
        DEC    SP
        PUSH   BC                          ; save arr top

        LD     H,L
        LD     L,0
GLN_LOOP:
        LD     A,(DE)                      ; key buf --/
        LD     (BC),A                      ;          /--> arr
        OR     A
        JR     Z,GLN_SKIP
        INC    DE
        INC    BC
        INC    L
        DEC    H
        JR     NZ,GLN_LOOP
        XOR    A
        LD     (BC),A                      ; str end coed '00'
GLN_SKIP:
        LD     H,0
        POP    BC
        LD     A,(BC)
        CP     1BH                         ; top = 1B ?
```

▶11月号167ページの岡村直樹さんへ。東村山音頭はいつも1,3,4丁目しか歌われていなかったと思いますよ。　鈴木　康夫(24)岐阜県

```
        RET     NZ
        LD      A,-1
        LD      (BC),A                  ; 1Bh -> -1
        LD      HL,-1                   ; cnt
        RET

```

```
;==================================================
;       linput( d, n )    by N.ito   '91/9/23
;==================================================

_GETL   EQU     1FD3H
_KBFAD  EQU     1F76H
_CSR    EQU     2018H

linput::
        CALL    _CSR
        LD      DE,(_KBFAD)
        CALL    _GETL
        LD      A,(DE)
        CP      1BH
        CALL    NZ,CHR_SKIP

        INC     SP                      ; skip return
        INC     SP
        POP     HL                      ; max
        POP     BC                      ; arr top
        PUSH    BC
        PUSH    HL
        DEC     SP
        DEC     SP
        PUSH    BC                      ; save arr top

        LD      H,L
        LD      L,0
LN_LOOP:
        LD      A,(DE)                  ; key buf --/
        LD      (BC),A                  ;           /--> arr
        OR      A
        JR      Z,LN_SKIP
        INC     DE
        INC     BC
        INC     L
        DEC     H
        JR      NZ,LN_LOOP
        XOR     A
        LD      (BC),A                  ; str end coed '00'
LN_SKIP:
        LD      H,0
        POP     BC                      ; load arr top
        LD      A,(BC)
        CP      1BH                     ; top = 1B ?
        RET     NZ
        LD      A,-1
        LD      (BC),A                  ; 1Bh -> -1
        LD      HL,-1                   ; cnt
        RET

CHR_SKIP:
        INC     L
CHR_SLOOP:
        DEC     L
        RET     Z
        INC     DE
        JR      CHR_SLOOP

```

```
/*================================================
        rnd(n)                  by N.ito   '91/9/28
================================================*/

#include        <stdio.h>

extern          rnd_sub();

rnd( n )
        int     n;
{
        int     x, sign;

        sign = 1;
        if ( n == 0 )
                x = 0;
        else
        {
                if ( n < 0 )
                {
                                n *= -1;
                                sign = -1;
                }
                n++;
                x = rnd_sub() & 0x7fff;
                x = x - ( x / n ) * n - 1;
                if ( x < 0 )
                                x++;
        }
        return( x * sign );
}

```

```
;==================================================
;       rnd_sub()               Oh!mz Random
;==================================================

rnd_sub::
        LD      DE,(OLDRND)
        LD      BC,(STEP)
        CALL    MULTI
        LD      (OLDRND),HL

        EX      DE,HL
        LD      HL,(HL_BUFF)
        LD      (HL),E
        INC     HL
        LD      (HL),D
        EX      DE,HL
        RET

MULTI:
        LD      HL,0
        LD      A,10H
MLOOP:
        ADD     HL,HL
        SLA     E
        RL      D
        JR      NC,SKIP
        ADD     HL,BC
SKIP:
        DEC     A
        JR      NZ,MLOOP
        RET

HL_BUFF:        DW      0
OLDRND: DW      OLDRND
STEP:   DW      899

```

```
;==================================================
;       beep()                  by N.ito   '91/9/22
;==================================================

_BELL   EQU     1FC4H

beep::
        CALL    _BELL
        RET

```

```
;==================================================
;       sgn( d )        by N.ito   '91/9/23
;==================================================

sgn::
        POP     BC
        POP     DE
        PUSH    DE
        PUSH    BC
        BIT     7,D
        JR      Z,SGN_SKIP
        LD      HL,-1                   ; n < 0
        RET
SGN_SKIP:
        LD      HL,1
        LD      A,D
        OR      E
        RET     NZ                      ; n > 0
        DEC     HL                      ; n = 0
        RET

```

```
;==================================================
;       sex( d )        by N.ito   '91/9/23
;==================================================

sex::
        POP     BC              ; skip return
        POP     HL              ; d
        PUSH    HL
        PUSH    BC
        BIT     7,L
        RET     Z
        LD      A,L
        NEG
```

▶X68000 EXPERTのフロッピーディスク挿入口にCDを入れたら取れなくなりそうになった。もうやめとこう。
竹永　昌伸(17)兵庫県

```
        LD      L,A
        LD      H,0FFH
        RET

;==============================================
;       bit( d, n )      by N.ito   '91/9/23
;==============================================

bit::
        POP     HL                      ; skip return
        POP     BC                      ; n
        POP     DE                      ; d
        PUSH    DE
        PUSH    BC
        PUSH    HL

        LD      A,0FH
        AND     C

        CP      8
        LD      B,E
        JR      C,BIT_SKIP1
        SUB     8
        LD      B,D
BIT_SKIP1:
        INC     A
        LD      C,A
        LD      A,1
BIT_LOOP:
        DEC     C
        JR      Z,BIT_SKIP2
        SLA     A
        JR      BIT_LOOP
BIT_SKIP2:
        LD      HL,0
        AND     B
        RET     Z
        INC     HL
        RET
```

```
;==============================================
;       set( d, n )      by N.ito   '91/9/23
;==============================================

set::
        POP     BC              ; skip
        POP     DE              ; d
        POP     HL              ; n
        PUSH    HL
        PUSH    DE
        PUSH    BC

        LD      A,0FH
        AND     E

        CP      8
        JR      C,SET_SKIP
        SUB     8
        CALL    SET_SUB
        OR      H
        LD      H,A
        RET
SET_SKIP:
        CALL    SET_SUB
        OR      L
        LD      L,A
        RET

SET_SUB:
        INC     A
        LD      E,A
        LD      A,1
SET_LOOP:
        DEC     E
        RET     Z
        SLA     A
        JR      SET_LOOP
```

```
;==============================================
;       reset( d, n )    by N.ito   '91/9/23
;==============================================

reset::
        POP     BC              ; skip
        POP     DE              ; d
        POP     HL              ; n
        PUSH    HL
        PUSH    DE
        PUSH    BC

        LD      A,0FH
        AND     E

        CP      8
        JR      C,SET_SKIP
        SUB     8
        CALL    SET_SUB
        XOR     0FFH
        AND     H
        LD      H,A
        RET
SET_SKIP:
        CALL    SET_SUB
        XOR     0FFH
        AND     L
        LD      L,A
        RET

SET_SUB:
        INC     A
        LD      E,A
        LD      A,1
SET_LOOP:
        DEC     E
        RET     Z
        SLA     A
        JR      SET_LOOP
```

```
;==============================================
;       peek(adr)          by N.ito   '91/9/22
;==============================================

peek::
        POP     BC
        POP     DE
        PUSH    DE
        PUSH    BC
        LD      A,(DE)
        LD      L,A
        LD      H,0
        RET
```

```
;==============================================
;       peekw(adr)         by N.ito   '91/9/22
;==============================================

peekw::
        POP     BC
        POP     DE
        PUSH    DE
        PUSH    HL
        PUSH    DE
        PUSH    BC
        LD      (HL),E
        RET
```

```
;==============================================
;       poke(adr)          by N.ito   '91/9/22
;==============================================

poke::
        POP     BC
        POP     DE
        POP     HL
        PUSH    HL
        PUSH    DE
        PUSH    BC
        LD      (HL),E
        RET
```

```
;==============================================
;       pokew(adr)         by N.ito   '91/9/22
;==============================================

pokew::
        POP     BC
        POP     DE
        POP     HL
        PUSH    HL
        PUSH    DE
        PUSH    BC
        LD      (HL),E
        INC     HL
        LD      (HL),D
```

▶「森の案内人」が見つからず，1年以上もほったらかしにしていた「ねじ式」がやっと解き終わりました。いまさらながらおもしろいですね。私はこれから映画「無能の人」を観て，しばらく人生をさまよってみようと思います。 野田　敏之(20)神奈川県

```
        RET

```

```
;=================================================
;       prmode(n)        by N.ito   '91/9/22
;=================================================

_LPTON  EQU    1FD9H
_LPTOF  EQU    1FD6H

prmode::
        POP    BC           ; Skip over return address
        POP    DE           ; mode
        LD     A,D
        PUSH   DE
        PUSH   BC
        OR     E
        CALL   Z,_LPTOF
        CALL   NZ,_LPTON
        RET

```

リスト2

```
/*========================================

        ｻﾝﾌﾟﾙ ( ﾓｸﾞﾗ ﾀﾀｷ )   By N.Ito

   --------------------------------------
    memo
        ｲｿｲﾃﾞ ﾂｸｯﾀ ﾉﾃﾞ ｱﾏﾘ ﾖｲ ｻﾝﾌﾟﾙ
        ﾄﾊ ｲｴﾅｲﾀﾞﾛｳ･･･｡

        ｱｿﾌﾞﾄｷﾊ ｵｵﾓｼﾞ ( CAPS ｦ ON )
        ﾆ ｼﾃｸﾀﾞｻｲ｡

        ﾎﾞｳｿｳ ｽﾙｶﾓ ｼﾚﾅｲ ﾉﾃﾞ ﾖｳﾁｭｳｲ｡
        ｷｰﾎﾞｰﾄﾞ ﾉ ﾐﾀﾞﾚｳﾁ ﾅﾄﾞﾊ
        ｾﾞｯﾀｲ ｷﾝﾓﾂ｡

========================================*/

#include       <stdio.h>

#define W_TIME            2000

main()
{
  int   sw, i, j;
  int   ichi, mae, hp, top;
  int   x[25],y[25],chr[25];
  char  top_n[15];

  widch(40); cls();
  top = 0;
  strcpy( top_n,"Oh! X" );
  sw = ask_play( top, top_n );
  while( sw )
  {
     cls();
     init( x, y, chr );
     hp = game_main( x, y, chr );
     pause( 10 );
     if ( hp >= top )
     {
        top = hp;
        inp_name( top, top_n );
     }
  cls();
  locate( 11, 12 ); printf("your = %3d percent",hp);
  sw = ask_play( top, top_n );
  }
  printf("¥n");
  exit();
}

/*==========   game main   ============*/

game_main( x, y, chr )
  int   *x, *y, *chr;
{
  int   i, j, hp, ht, sw;
  int   mae, ichi, hum, key;

  hp = ht = hum = 0;
  for( i = 1; i <= 30; i++ )
  {
     while( TRUE )
     {
           ichi = rnd(24);
           if ( ichi != mae ) break;
     }
  mae = ichi;
  locate( x[hum ], y[hum ] - 1 ); printf("  ");
  locate( x[ichi], y[ichi] ); printf("&");
  sw = 1;
  for( j = 0; j < W_TIME * 2; j++ )
  {
     key = inkey( 0 );
     if ( sw == 1 && key != 0 )
     ht += check( key, ichi, x, y, chr, &hum, &sw );
  }
   locate( x[ichi], y[ichi] ); printf(" ");
   hp = ( ht * 100 ) / i;
   locate( 11, 20 ); printf("ﾒｲﾁｭｳﾘﾂ %3d percent",hp );
  }
  return( hp );

/*----------   check        ------------*/

check( key, ichi, x, y, chr, hum, sw )
  int   key, ichi, *x, *y, *chr, *hum, *sw;
{
  int   i, ht;

  locate( x[ *hum ], y[ *hum ] - 1 ); printf("  ");
  if ( key == chr[ ichi ] )
  {
     ht   = 1;
     *sw  = 0;
     *hum = ichi;
     locate( x[ *hum ], y[ *hum ] - 1 ); printf("#-");
     locate( x[ *hum ], y[ *hum ]     ); printf("*");
     beep();
  }
   else
  {
    ht = 0;
    for( i = 0; i < 24; i++ )
    {
      if ( chr[ i ] == key )    break;
    }
      if ( i < 24 )
      {
       *hum = i;
         locate( x[ *hum ], y[ *hum ] - 1 ); printf("#-");
      }
    }
 return( ht );
}



/*==========   start?    ============*/

ask_play( top, top_n )
  int   top;
  char  *top_n;
{
  locate( 8, 4); printf("<<<<<   ﾓｸﾞﾗ ﾀﾀｷ   >>>>>");
  locate(11, 8); printf("top  = %3d percent",top);
  locate(11,10); printf("name = %s",top_n);
  locate(14,16); printf("Play Y / N  ");
  return( inkey(1)=='Y' ? 1 : 0 );
}

/*==========  initialize   ============*/

init( x, y, c )
  int   *x, *y, *c;
{
  int   i, j, k;
  int   *cc;

  cls();
  locate( 8, 2); printf("<<<<<   ﾓｸﾞﾗ ﾀﾀｷ   >>>>>");

  cc = c;
  for( i = 0; i < 3; i++ )
  {
    for( j = 0; j < 8; j++ )
    {
         *x = j * 4 + 6;
         *y = i * 4 + 6;
         *c = chr_set( cc );
   locate( *x - 1, *y + 1 ); printf("[%c]",*c++ );
         x++; y++;
    }
  }
}

/*----------- character set ------------*/

chr_set( c )
  int   *c;
{
  int   i, chr, sw;
  sw = 1;
  while( sw )
  {
     chr = rnd(26) + 'A';
     for( i = 0; i < 26; i++ )
     {
        if ( *(c + i) == chr )
             {
                sw = 1;
                break;
             }
        else
```

▶祝！ Oh!X平積み化。いやあ，近所の本屋がついにOh!Xを平積みにしたんです。もうこれからは探すのに苦労しないぞ。
小林　敦(16)埼玉県

```
                sw = 0;
        }
    }
        return( chr );
}

/*==========    input name  ============*/

inp_name( top, top_n )
  int   top;
  char  *top_n;
{
  cls();
  locate( 8, 4); printf("<<<<<   ﾓｸﾞﾗ ﾀﾀｷ   >>>>>");
  locate(11, 8); printf("You are champion !!");
  locate(11,10); printf("ﾒｲﾁｭｳﾘﾂ %3d percent",top );
  locate(13,13); printf("Input yor name");
  locate(11,16); printf("name = ");
  linput( top_n,12 );
}

/*==========    pause       ============*/

pause( n )
  int   n;
{
  int   i;
  for( ; n > 0; n-- )
    {
      for( i = W_TIME; i > 0; i-- ) ;
    }
}
```

リスト3

```
int input,output;
int argcs;
int files,eof;
char *lines,pline[];
extern int *argvs;

#define stdin 0
#define stdout 1
#define YES 1
#define NO  0
#define EOF -1
#define NOCCARGC
/*
** input and output file opens
*/
openfile() {     /* extire function revised 39 */
 char fn[20];    /* file name buffer 2+13+1+3+1*/
 char *c,dptr;   /* command line pointer */
 int i,ext;
 input = EOF;
 line = pline;

 while(++filearg < argcs) {
   cmdptr = argvs[filearg];
   if(cmdptr[0]=='-') continue;

   ext = NO;
   i = 0;

   strncpy(fn,cmdpt,15); /* Debugged '91 Oct/16th */

   while(cmdptr[i] && i < 15) {
      if(cmdptr[i] == '.') {
         ext = YES;
         break;
      }
      i++;
   }
   if(!ext) strcpy(fn + i,".C");

   input = mustopen(fn,"r");

   if(!files && isatty(stdout)) {
      strcpy(fn + i,".ASM");
      output = mustopen(fn,"w");
   }
   files=YES;
   kill();
   return;
 }
 if (files++) eof=YES;
 else input=stdin;
 kill();
 }
```

リスト4

```
3EFD 21 E6 FF 39 F9 21 FF FF : 57
3F05 22 F0 38 2A 42 38 22 46 : 56
3F0D 38 2A 06 39 23 22 06 39 : 25
3F15 EB 2A C2 38 CD D9 A6 7C : D7
3F1D B5 CA 5E 40 21 04 00 39 : 7B
3F25 E5 2A C4 38 EB 2A 06 39 : 5F
3F2D 29 CD 57 A6 CD 9B A6 21 : 22
3F35 04 00 CD 5B A6 CD 51 A6 : 96
3F3D EB 21 2D 00 CD B9 A6 7C : E1
3F45 B5 CA 4C 3F C3 0E 3F 21 : 3B
3F4D 00 00 39 EB 21 00 00 CD : 12
3F55 9E A6 21 02 00 39 EB 21 : AC
3F5D 00 00 CD 9E A6 21 06 00 : 38
3F65 39 E5 21 06 00 CD 5B A6 : 13
3F6D E5 CD D1 A3 C1 C1 21 04 : CD
3F75 00 CD 5B A6 EB C1 E1 E5 : 40
---------------------------------
SUM: 89 FB 32 66 AD 5A FD 4D FC1E

3F7D C5 CD 4A A6 7C B5 CA 9B : 18
3F85 3F C1 D1 D5 C5 21 0F 00 : 9B
3F8D CD D9 A6 7C B5 CA 9B 3F : 21
3F95 21 01 00 C3 9E 3F 21 00 : E3
3F9D 00 7C B5 CA D5 3F 21 04 : 34
3FA5 00 CD 5B A6 EB C1 E1 E5 : 40
3FAD C5 CD 4A A6 EB 21 2E 00 : BC
3FB5 CD B9 A6 7C B5 CA CB 3F : 31
3FBD 21 00 00 39 EB 21 01 00 : 67
3FC5 CD 9E A6 C3 D5 3F 21 02 : 0B
3FCD 00 CD 8E A6 2B C3 73 3F : A1
3FD5 E1 E5 CD 28 A7 7C B5 CA : 5D
3FDD F3 3F 21 06 00 39 EB C1 : 3E
3FE5 E1 E5 C5 19 E5 21 83 40 : 6D
3FED E5 CD D1 A3 C1 C1 21 06 : CF
3FF5 00 39 E5 21 86 40 E5 CD : B7
---------------------------------
SUM: 0C B1 5E F9 B2 C4 4E E1 31D6

3FFD 9A 40 C1 C1 22 F0 38 2A : D0
4005 04 39 CD 28 A7 7C B5 CA : D4
400D 22 40 21 01 00 E5 CD 8F : C5
4015 A3 C1 7C B5 CA 22 40 21 : E2
401D 01 00 C3 25 40 21 00 00 : 4A
4025 7C B5 CA 4F 40 21 06 00 : B1
402D 39 EB C1 E1 E5 C5 19 E5 : 6E
4035 21 88 40 E5 CD D1 A3 C1 : D0
403D C1 21 06 00 39 E5 21 8D : B4
4045 40 E5 CD 9A 40 C1 C1 22 : 70
404D 02 39 21 01 00 22 04 39 : BC
4055 CD 45 63 21 1A 00 39 F9 : E2
405D C9 2A 04 39 23 22 04 39 : B2
4065 2B 7C B5 CA 74 40 21 01 : FC
406D 00 22 EE 38 C3 7A 40 21 : E6
4075 00 00 22 F0 38 CD 45 63 : BF
---------------------------------
SUM: FE EE D9 C0 EA BC 85 E9 9333

407D 21 1A 00 39 F9 C9 2E 43 : A7
4085 00 72 00 2E 41 53 4D 00 : 81
408D 77 00 20 44 53 20 00    : 4E
---------------------------------
SUM: 98 8C 20 AB 8D 3C 7B 43 DC38
```

リスト5

```
/*
** ungetc.c     by fas 16/10/91
*/

#include <stdio.h>
#include <clib.def>

extern char __con;

ungetc(c,fd) char c, *fd;{

        if(fd == stdin) {
                if(__con) return(EOF);
                return(__con = c);
        }
        if(fd[UNGOT]) return(EOF);
                fd[UNGOT] = c;
        return(c);
}
```

★(で)のショートプロぱーてぃ その27

★よもう一度！

Komura Satoshi 古村　聡

1990年9月号に掲載された「なさけない★星★」がパワーアップして帰ってきた。もう「なさけない★星★」とは呼ばせない？　ということで，今月はタイプ練習「KPP.BAS」，そして，「なさけない★星★ PART2」の2本を紹介します。

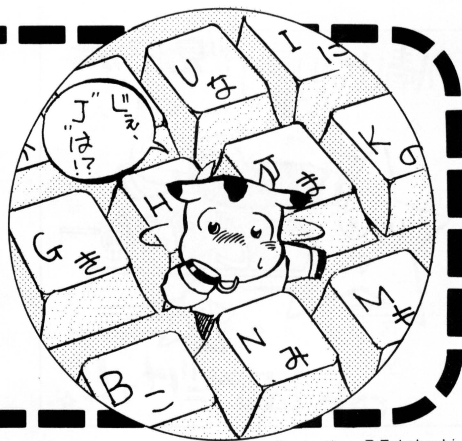


illustration : T.Takahashi

変な話なんですが，最近私，プログラムに燃えてます。

ここしばらく，例のZ-MUSICシステムの別冊の仕事とやらで，善ちゃんたちが必死になってプログラム組んだりしてるんですが，いやいや，なかなか苦しんでおいでのようです。

しかし，他人が「あ～，ここが2バイト縮められるぅ」なんて苦しんでいるんだから，こっちは「いいキミじゃのう。キヒヒヒヒヒ」とイヂワルヂイサンしてればいいものを……。ああ，あろうことかあるまいか，なぜか，「ああっ，プログラムがしたいっ！ 脳ミソがとろけるほどデバッグがしたいしたいしたいっ！」などと思ってしまうのです。

それでいて，いざ自分のプログラムのデバッグを始めると「ああっ，こんなもん作るんじゃなかった！　投げちゃおっかな」とかやってんだから困ったもの。

しかしですねぇ。パソコン持っていて，何がいちばん楽しいって，ゲームでもなけりゃあ，ワープロで原稿書くことでもなく，やっぱりプログラムをすることなんじゃないでしょうかねぇ。プログラム上に成り立つ自分だけの世界よ……ああっ！

ということで，今月もレッツ・プログラミングなのだっ！　ああっ，なんて，まともな言い回しなんだ。



燃えるキーボード

では，最初のプログラム。今月1本目は千葉県の林さんの作品でX-BASIC用のキータイプ練習プログラム，「KPP.BAS」です。

KPP.BAS for X68000

(X-BASIC)

千葉県　林　順一

KPPとはKDDでもなければみかか(NTTのことね）でもない？ じゃなくて，KPP.BASのKPPはKey Practice Program,早い話がキーボード練習プログラムなのです。

このプログラムはレベル1とレベル2の2つのモードがありまして，レベル1のほうはA～Zまでをとにかく早く打っていく練習。レベル2のほうはアルファベットがランダムに表示されていくので，そのキーを打っていく，というゲームになっています。

ABCDEFG……，えーいっ，Vはどこだっ，Wはっ!?　前にX1用のキーボード訓練プログラムが（上からアルファベットが降ってくるヤツね）あったけど，今回はそれよりもきびしいぞぉ！ なにしろあれの場合はインベーダーとの戦いだったけど，今回は自分との戦いなのだから。

ついに私，このたびレベル2で－361を記録してしまいました。ふっふっふっ。いやあ，やっぱり毎月原稿を書いているおかげですかねぇ（毎日のようにチャットをやってたせいじゃないかっていう気もするんだけど……。みかか代がかかってるせいか意地でもキータイプが早くなる)。

にしても，投稿ディスクの中の，レベル1の1448ってスコアはいったいなんなんじゃあ！ うう，どうしてもVから先がわからなくなる自分がうらめしい……。目的からいえばこれって実用プログラム（教育プログラム？）なんだろうけど，なんだかなー。だんだんゲームプログラムのような気がしてくる自分がこわい。



KPP.BAS



なさけない★星★ PART2

考えてみれば教育プログラムだって面白ければ，ゲームプログラムと変わらないんだよねぇ。教育ソフトってつまんなさそうだけど（実際にやったことはないからわからないけど)，どうなんでしょうね？ 本当に教育ソフトが面白ければ，勉強も遊びも変わらなくなるのにねぇ。ああ，もっと早くパソコンやらファミコンやらが出てて，もっと教育ソフトも面白ければ，いまごろ私の成績も……。いえいえ，なんでもありましぇーん。



帰ってきた★なのだ！

というわけで，さくさくと次のプログラムいきます。今月2本目のプログラムはX68000用の環境プログラム，「なさけない★星★ PART2」なのです！

なさけない★星★ PART2 for X68000

(X-BASIC)

埼玉県　遠藤　克之

BASICによる環境ソフト（……と呼べるかどうかはチトあやしい気もする）なのであります。

実行方法は簡単。BASICを立ち上げてプログラムを実行したらカラフルな星が表示されます。ジョイスティックを上下左右に動かすと星が動きます。というより，宇宙空間の中を自分がくるくる回るという感じかな。えー，ほかに説明のしようがないな



ぁ。そういうソフトです。うん。

な，なっつかしぃーっ！「なさけない★星★」ですねっ。思わず，バックナンバーを引っ張り出してしまったじゃあないですか！　えーっと，1990年９月号，132ページの“ショートプロぱーてぃ その13 なさけなくない星？”なのであります。初代の「なさけない★星★」は「せっかくＣを買ったからＣで組んで」あったんですよね。今回はBASICだけど。Ｃコンパイラはお元気ですか？　関係ないけど，その月はぱーてぃハンズ第１部の最終回だったのです。今回は第３部の最終回。なにか因縁めいたものすら感じてしまうぜ。

ちなみにそのころ発売だったゲームっていうと，「シムシティー」でしょ，ハムスライスのかあいい「ワールドコート」でしょ，それから「闇の血族」があって，それに「第４のユニット５　D-Again」。……ううっ，１年たつと昔の話をしているような気がしてくるとは，こわい業界。これじゃ，いなくなったらあっという間に忘れさられてしまうな……。遠藤さんも忘れられないように，がんがんPART3でもPART4でも作ってくださいな。おお，そうだ。

「なさけない★星★ PART2」は「なさけない★星★」がなくても楽しむことができます（ゲーム広告のまねっ！）。

しかし，このプログラムはゲームなんかにも使えそうですねぇ。キャラクターを全部スプライトにして，プログラムはコンパイルするようにすれば，４方向シューテングとかに使えるんじゃないでしょうか。だれかやってみる気ない？

ああ，今月も話が脱線しまくってしまった。いいんだろうかこんなことで。ショートプロぱーてぃなのに……。

ということで，一抹の不安を残しつつ，また来月。うーむ……。

リスト１　KPP.BAS

```
  10 /*kpp.bas
  20 /*   by J.Hayashi
  30 /*          1991/10/3
  40 screen 0,1,1,1
  50 int sc,hs(2,1),level,replay,ch(25),ai,bi,ci
  60 str st,hsn(2,1),d[256]
  70 op()
  80 back()
  90 fle():hsc()
 100 repeat                                /*全体のリピート
 110 replay=0:sc=2000                      /*コンパイルする場合はSC=10000
 120 locate 5,11:print "INPUT LEVEL(1-2)"
 130 repeat
 140 st=inkey$
 150 until (st="1") or (st="2")
 160 locate 5,11:print "                "
 170 level=atoi(st)-1
 180 for i=0 to 25
 190 ch(i)=i+65
 200 next
 210 if level=1 then {
 220 for i=0 to 100                        /*シャッフル
 230 ai=rnd()*25
 240 bi=rnd()*25
 250 ci=ch(ai)
 260 ch(ai)=ch(bi)
 270 ch(bi)=ci
 280 next
 290 sc=2000                                /*ここもSC=40000
 300 }
 310 locate 5,11:print "HIT ANY KEY"
 320 locate 10,12:print "TO START!!"
 330 st=inkey$
 340 locate 5,11:print "           "
 350 locate 10,12:print "          "
 360 for i=0 to 25                         /*メイン
 370 locate 8,8:print chr$(ch(i))
 380 repeat
 390 st=inkey$(0):strupr(st)               /*inkey$(0) キー入力を待たない
 400 sc=sc-1
 410 until st=chr$(ch(i))
 420 sc=sc+1:beep
 430 next
 440 locate 24,8:print sc
 450 locate 8,8:print " "
 460 repeat:st=inkey$(0):until st<>""
 470 ai=hschk()                            /*ハイスコア?
 480 if ai=0 then {
 490 hs(2,level)=hs(1,level)
 500 hs(1,level)=hs(0,level)
 510 hs(0,level)=sc
 520 hsn(2,level)=hsn(1,level)
 530 hsn(1,level)=hsn(0,level)
 540 inname()
 550 hsn(0,level)=st }
 560 if ai=1 then {
 570 hs(2,level)=hs(1,level)
 580 hs(1,level)=sc
 590 hsn(2,level)=hsn(1,level)
 600 inname()
 610 hsn(1,level)=st }
 620 if ai=2 then {
 630 hs(2,level)=sc
 640 inname()
 650 hsn(2,level)=st
 660 }
 670 cls
 680 hsc()
 690 fse()
 700 locate 5,11:print "REPLAY (Y OR N)"
 710 repeat
 720 st=inkey$:strupr(st)
 730 if st="Y" then replay=1
 740 if st="N" then replay=2
 750 until (replay=1) or (replay=2)
 760 locate 5,11:print "               "
 770 until replay=2
 780 end                                   /*end
 790 func hsc()                            /*ハイスコア表示
 800 for j=0 to 1
 810 for k=0 to 2
 820 locate j*14+3,k+2:print hsn(k,j)
 830 st=right$("        "+str$(hs(k,j)),5)  /*スコアを右に揃える
 840 locate j*14+11,k+2:print st
 850 next
 860 next
 870 endfunc
 880 func inname()                         /*名前登録
 890 locate 5,11:print "<< HI-SCORE >>"
 900 st=inkey$
 910 locate 5,11:print "INPUT NAME (8)"
 920 locate 10,12:input st
 930 st=st+"        "                      /*"  "は8文字
 940 mid$(st,0,8)                          /*8文字以上カット
 950 endfunc
 960 func hschk()                          /*ハイスコアのチェック
 970 for i=0 to 2
 980 if sc>=hs(i,level) then return(i)
 990 next
1000 return(3)
1010 endfunc
1020 func fle()                            /*ファイルの読み込み
1030 error off
1040 ai=fopen("kppsco","r") :error on
1050 if ai=-1 then {
1060 for l=0 to 1
1070 for m=0 to 2
1080 hsn(m,l)="someone "
1090 hs(m,l)=-9999
1100 next
1110 next
1120 } else {
1130 d="":freads(d,ai):fclose(ai)
1140 for l=0 to 1
1150 for m=0 to 2
1160 hsn(m,l)=mid$(d,l*24+m*8+1,8)
1170 hs(m,l)=atoi(mid$(d,l*15+m*5+49,5))
1180 next
1190 next
1200 }
1210 endfunc
1220 func fse()                            /*ファイルにセーブ
1230 d=""
1240 for l=0 to 1
1250 for m=0 to 2
1260 d=d+left$(hsn(m,l)+"        ",8)
1270 next
1280 next
1290 for l=0 to 1
1300 for m=0 to 2
1310 d=d+left$(str$(hs(m,l))+"     ",5)
1320 next
1330 next
1340 ai=fopen("kppsco","c")
1350 fwrites(d,ai)
1360 fclose(ai)
1370 endfunc
1380 func back()                           /*背景の設定
1390 symbol(75,0,"HI-SCORE",1,1,1,13,0)
1400 symbol(25,15,"LEVEL 1",1,1,1,11,0)
1410 symbol(150,15,"LEVEL 2",1,1,1,11,0)
1420 symbol(120,210,"PRODUCED BY",1,1,0,6,0)
1430 symbol(135,225,"JUNICHI-HAYASHI",1,1,0,6,0)
1440 symbol(88,128,"YOUR SCORE =",1,1,1,15,0)
```

```
1450 symbol(0,32,"1",1,1,1,15,0)
1460 symbol(0,48,"2",1,1,1,15,0)
1470 symbol(0,64,"3",1,1,1,15,0)
1480 box(62,127,72,143,15,65535)
1490 box(61,126,73,144,14,65535)
1500 box(60,125,74,145,1,65535)
1510 endfunc
1520 func op()                                    /*オープニングの設定
1530 screen 0,1,1,1
1540 symbol(0,0,"Key",2,2,2,3,0)
1550 symbol(25,50,"Practice",2,2,2,9,0)
1560 symbol(50,100,"Program",2,2,2,5,0)
1570 symbol(75,175,"HIT ANY KEY !!",1,1,1,11,0)
1580 symbol(100,200,"by J.Hayashi",1,1,2,7,0)
1590 locate 30,15
1600 repeat
1610 st=inkey$(0)
1620 ai=rnd()
1630 until st<>""
1640 screen 0,1,1,1
1650 endfunc
```

リスト2 なさけない★星★ PART2

```
 100 /*         なさけない星 PART2
 110 /*                         PRODUCE BY K.ENDO ( 7K1*** )
 120 screen 0,1,1,1:window(0,0,511,511):randomize(val(right$(ti
me$,2))*100)
 130 int sx,sy,s,s1,i
 140 dim int x(3),y(3)
 150 float p
 160 for i=0 to 3:apage(i):home(i,0,0):for st=0 to 25*(i+1)
 170    set(rnd()*511,rnd()*511,rnd()*15):next:next:vpage(15)
 180 /*-=-=-=-=-=-=-=-=-=-=       MAIN       =-=-=-=-=-=-=-=-=-=-
 190 while 1:s=stick(1):if s<>0 then s1=s
 200    if s then p=p+0.3#*-(p<16) else p=p-0.3#*-(p>0):if p<0
then p=0
 210    sx=((s1=4 or s1=7 or s1=1)-(s1=6 or s1=9 or s1=3))*p
 220    sy=((s1=8 or s1=7 or s1=9)-(s1=2 or s1=1 or s1=3))*p
 230    for i=0 to 3:x(i)=x(i)+sx/(i+1)*2:y(i)=y(i)+sy/(i+1)*2
 240       if x(i)<0 then x(i)=511 else if x(i)>511 then x(i)=0
 250       if y(i)<0 then y(i)=511 else if y(i)>511 then y(i)=0
 260       home(i,x(i),y(i))
 270 next:endwhile
 280 end
 290 /*-=-=-=-=-=-=-=-=-=-=       SUB       =-=-=-=-=-=-=-=-=-=-
 300 func set(c,d,e)
 310    circle(c,d,(4-i)/2,e):paint(c,d,e)
 320 endfunc
```

(で)のぱーてぃハンズ第3部――(最終回)

前回まででミニマックス法&選択二十五のプログラムの話は終わったんですよね。お疲れさまでした。プログラムはちゃんと動きましたか？　簡単な作りにしては結構遊べる思考ルーチンになってるでしょ。

ではでは，今月は一応，思考ゲームのプログラムに欠かせない（でも今回のプログラムではやっていない）α-β枝刈りについてのお話なぞをしてみます。

先月まで説明してきたミニマックス法は，なにしろ片っ端から打てる手が有利かどうかを調べていくわけで，いかに人間様より計算の速いパソコンとはいっても，やっぱりそれなりに深い読みをしようとすればするほど，がんがんと加速度的に時間を費やしてしまうようになるのであります。

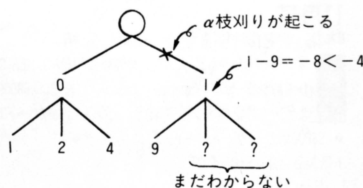
しかしながら，昔の偉い人はこのゲーム木をじーっと見ながら考えた。こりゃあ，なんとかしてもうちょい思考時間を減らせるんじゃないかと。そこで考えたのが枝刈りというものなのであります。

昔の人は思ったんですね。このゲーム木にはやっぱり無駄が多いんじゃないかと。考えてみりゃあ，もし，片っ端から調べないで，一発でいちばんいい手にたどりつければ，枝のたくさんある木なんか作らないですむ。それじゃあ，一発ってわけにはいかないだろうけど，少しでも無駄な枝を作らないようなゲーム木を作るような方法を開発しよう。枝の少ないゲーム木を作る，すなわち枝刈りをするのであります。

それからさまざまな人によってさまざまな枝刈りの方法が開発されました。そのなかでも2大有名枝刈りというのが，陽の南斗聖拳，陰の北斗神拳，じゃなくて，広さ優先のSSS法と深さ優先のα-β枝刈り法なのです。

当然，北斗の拳と同じく，陰のα-β枝刈り法のほうが有名なのはあらためていうまでもないかな？

図



奥義，α-β!

では，実際に木を書いてα-β枝刈りがどのような動きをするのか見てみましょう。

α-β枝刈り法というのは，基本的な動きはミニマックス法の動きと同じです。図を見てください。例によって評価値の書かれたゲーム木なのであります。

さて，最初に自分の1番目の手（0点の手ですね）を見にいきます。そして，次に自分がその手を打った場合の敵の打ってくる手について，片っ端から考えるわけです。1点，2点，4点，と。敵はいちばん大きな手である4点の手を取ってくるでしょうね。ここで手の先読みを終わりにするとすれば，自分が1番目の手を打った場合の評価値は0－4＝－4点ということになります。

ここまではミニマックス法そのものですね。枝もまったく減っていません。

では，自分の次の手について考えてみましょう。2番目の手は1点です。で，その場合の9点……おおっと，ストップ！　9点ですね。そう9点なんです。

この時点でこの9点の手について考えてみましょう。もし，残り2つの手が9点より小さかった場合，敵にとって9点は最大の点数ですから，敵はこの9点の手を取ってくるでしょう。その場合，自分の2番目の手の評価値は，

$$1-9=-8$$

になるわけです。

そして，もし敵の残りの手が9点以上だった場合。敵はそっちの手を取ってきますね。必ず9点よりも大きいわけですから，ここでは9＋n点としておきましょう。その場合の評価値は

$$1-(9+n)=-8-n$$

です。

どっちにしても「－8点よりは低い」点数であるわけです。

さてさて，先ほどの1番目の手の評価値は－4点ですよね。2番目の手は「どうあがいても－8点以上いかない」んだから，絶対に1番目の手のほうが有利なのです。つ，ま，り，この2番目の手についてはこれ以上枝を作っても無駄，なんですよね。

そこでこの2番目の手を“ぷちっ！”と刈ってしまう（これ以上考えない）のです。これこそが「α枝刈り」というやつなのです。同様に敵から見て……とやる「β枝刈り」と合わせて，奥義「α-β枝刈り」となるのであります。

終わった終わった

ということなのでありました。「SSS法」についての詳しいところは自分で調べてね，ということにしてしまいます。

ところでなんで，枝刈りのやり方までやっておいて，なんでプログラムに使わなかったんでしょうか？

よく考えればわかるんですが，枝刈りするにはそれなりに条件判断が必要になるわけです。しかし，2手や3手しか先読みしないのでは条件判断に時間がかかりすぎて，枝刈りの意味が全然ないんですね。まして，選択二十五は1回に選べる手が少ないから，枝刈りしても刈れる手が少ない。ということで，省略してしまいました。決して手抜きじゃないんだぞ，と。

さて，ぱーてぃハンズ第3部，終わりました。いかがでしたでしょうか。本当に初歩の初歩だけど，思考ゲームってのがどんなものか，なんとなくわかってきたんじゃないかと思います。もう少し段階的にリストを発表していきたかったんですが，結果的に解説が多くて大変だったかもしれませんね。もし，機会があったら皆さんも自分の思考ゲームを作ってみてください。自分でも遊べるから楽しいですよ。

ああ，最後はヨタ話をするつもりだったのにマジメな話になってしまった。

それでは皆さん，ごきげんよう。

BACK ISSUES バックナンバー案内

ここには1990年12月号から1991年11月号までをご紹介しました。現在1990年10, 1991年1, 7～11月号の在庫がございます。バックナンバーおよび定期購読の申し込み方法については,172ページを参照してください。

1990



12月号（品切れ）
特集　XCのための傾向と対策

連載　X-BASICプログラミング調理実習/ハードウェア工作入門/マシン語プログラミング/ショートプロぱーてぃ/Z80's Bar/大人のためのX68000/ようこそここへC言語/INTEGRAL XI

●シミュレーションプログラミング入門
●特別企画アナログジョイスティックの制作

LIVE in '90　グラディウスIII/メタルサイト

THE SOFTOUCH SPECIAL　イメージファイト/ジェミニウイング/NAIOUS他

全機種共通システム　STACKコンパイラ

1991



1月号
特集　急接近! SX-WINDOW

特別付録　謹賀新年PRO-68K（5″2HD）

連載　ハードウェア工作入門/シミュレーションプログラミング入門/DōGA・CGA/ショートプロぱーてぃ/大人のためのX68000/PurePASCAL/清水和人流プログラミング道場/X-BASIC調理実習

LIVE in '91　めぞん一刻/涙で綴るパパへの手紙

THE SOFTOUCH　ソル・フィース/銀英伝II/続ダンジョン・マスター他

製品紹介　光磁気ディスクCZ-6MOI

全機種共通システム　ブロックアクションゲームCOLUMNS



2月号（品切れ）
特集1　グラフィックの〝実験的″手法

特集2　SX-WINDOWプログラミング

連載　ハードウェア工作入門/シミュレーションプログラミング入門/マシン語プログラミング/大人のためのX68000/Z80's Bar/ショートプロぱーてぃ/INTEGRAL XI/ようこそここへC言語

●1990年度 GAME OF THE YEARノミネート発表

LIVE in '91　Misty Blue/スプーンおばさん

THE SOFTOUCH　栄冠は君に/KLAX/ダイナマイト・デューク他

全機種共通システム　ダイスゲームKISMET



3月号（品切れ）
特集　MIDI & MUSIC PROCESSING

連載　ハードウェア工作入門/シミュレーションプログラミング入門/マシン語プログラミング/大人のためのX68000/Z80's Bar/ショートプロぱーてぃ/DōGA・CGA/C言語/PurePASCAL

●SXLIFE完結編/ウィンドウシステム大比較
●周辺機器新製品紹介

LIVE in '91　戦いの兜/LITTLE WING/リゾ・ラバ/花

THE SOFTOUCH　アトミック・ロボキッド/スペースローグ他

全機種共通システム　アクションゲームMUD BALLIN'



4月号（品切れ）
特集　人とゲームのインタフェイス

連載　DōGA・CGA/シミュレーションプログラミング入門/ハードウェア工作入門/ようこそここへC言語/Z80's Bar/ショートプロぱーてぃ/清水和人流プログラミング道場

●新連載　吾輩はX68000である/よいこのSX-WINDOW講座
●決定!　1990年度GAME OF THE YEAR

LIVE in '91　Easy Come, Easy Go!/シシリエンヌ

THE SOFTOUCH　メルヘンメイズ/中華大仙/スライス他

全機種共通システム　SLANG用カードゲームDOBON



5月号
特集　新登場!　X68000XVI/XVI-HD

特別付録　黄金週間PRO-68K（5″2HD）

第6回　言わせてくれなくちゃだワ

連載　ハードウェア工作/ようこそここへC言語/大人のためのX68000/X68000マシン語プログラミング/ショートプロぱーてぃ/マシン語カクテル in Z80's Bar

LIVE in '91　ブービーキッズ/NO.NEW YORK

THE SOFTOUCH　マーブル・マッドネス/シグナトリー/石道他

全機種共通システム　実数型コンパイラ言語REAL



6月号（品切れ）
特集　初心者のための環境構成術

創刊9周年記念Oh!Xアンケート結果大分析大会その1

連載　ハード工作/大人のためのX68000/Z80's Bar/DōGA/ようこそC言語/ショートプロぱーてぃ/SX-WINDOW/吾輩はX68000である/マシン語プログラミング

●響子 in CGわ～るど

LIVE in '91　暴れん坊将軍/ナディア/POWER HALL他

THE SOFTOUCH　パロディウスだ!/遥かなるオーガスタ/ノスタルジア他

全機種共通システム　S-OS 6周年記念　Small-C　処理系の移植



7月号
特集　Personal Tool,BASIC

別冊付録　X-BASIC　ポケットリファレンスブック

連載　大人のためのX68000/ハード工作/響子 in CGわ～るど/ショートプロぱーてぃ/SX-WINDOW/吾輩はX68000である/ようこそC言語/Z80's Bar/マシン語プログラミング

●X1用ゲーム　The Master of Payment

LIVE in '91　今すぐKISS ME/歩いていこう

THE SOFTOUCH　パロディウスだ!/ファランクス/スコルピウス/AIII他

全機種共通システム　実数型コンパイラ言語REAL ソースリスト編



8月号
特集　印刷の世界へ

連載　大人のためのX68000/SX-WINDOW/ようこそC言語/響子 in CGわ～るど/ハード工作/ショートプロぱーてぃ/吾輩はX68000である/マシン語プログラミング

●X68000カードゲーム　七並べ
●X1用ゲーム　DEFEAT2

LIVE in '91　パワードリフト/イースIII/TURBO OUTRUN

THE SOFTOUCH　黄金の羅針盤/サイレントメビウス/パロディウスだ!他

全機種共通システム　Small-C ライブラリの移植



9月号
特集　Brush up your MAGIC.

連載　マシン語プログラミング/DōGA/Z80'sBar/ショートプロ/響子 in CGわ～るど/ハード工作/シミュレーション入門/吾輩はX68000である/大人のためのX68000/C言語

●X1用ゲーム Manual Runner
●ANOTHER CG WORLD

LIVE in '91　One/WHITE MANE

THE SOFTOUCH　イース/生中継68/アークス・オデッセイ他

全機種共通システム　SLANG用NEWファイル入出力ライブラリ



10月号
特集　マシン語との邂逅

連載　響子 in CGわ～るど/マシン語プログラミング/ショートプロ/ハード工作/Z80'sBar/よいこのSX-WINDOW/ANOTHER CG WORLD/吾輩はX68000である/ようこそC言語/大人のためのX68000

●新連載　Computer Music入門
●NEW Print Shop PRO-68K Ver 2.0

LIVE in '91　うれしい!　たのしい!　大好き/SPANISH BLUE

THE SOFTOUCH　ボナンザブラザーズ/ロードス島戦記/ジーザスII他

全機種共通システム　Small-C活用講座（初級編）



11月号
特集　空間彷徨型ゲーム大分析

連載　響子 in CGわ～るど/大人のためのX68000/ANOTHER CG WORLD/DōGA/ショートプロ/Computer Music入門/吾輩はX68000である/ようこそC言語/マシン語プログラミング/Z80'sBar/ハード工作

●X68000用カードゲーム　ギャップ
●新製品紹介　F-Card GT

LIVE in '91　オーダイン

THE SOFTOUCH　キャメルトライ/アクアレス/フューチャーウォーズ他

全機種共通システム　Small-C活用講座（応用編）/MORTAL

自民党宮沢総裁誕生，西武―広島の日本シリーズと話題が盛り沢山だった10月下旬だが，世の中の話題を独占したのは，とんだ伏兵，宮沢りえちゃんだった。20日日曜日に読売，21日月曜日に朝日と２日連続で全国紙に前代未聞のヌード写真つきの写真集全面広告が掲載。芸能マスコミにとどまらず，右へ左への大騒ぎ。写真集の出版社では予約電話が殺到して対応できないとか，電話回線がパンクしそうになってNTTがあわてて対策を打ったとか，書店の組合があわてて抗議したとか，これにまつわる話もワンサカ。

「ヘア露出問題」を契機に，ヌード写真集は今年の出版界の話題の中心となっていた。樋口可奈子に続いて小柳ルミ子の写真集が爆発的な話題となっていたのは，ほんのわずか前のこと。こちらを軽くふっとばすように，突如ふってわいたのが，りえちゃん旋風だ。

よくよく考えてみると，わずか１枚の胸を露出した写真が広告に掲載されているだけ。しかも，熟女・ルミ子さんのような過激な期待ができるわけではないことは，あきらかだ。単にアイドルタレントのヌード写真集が発売されるというだけの話なのである。だが，ただそれだけの話でも，素材が人気ナンバーワン（というには疑問があるのだが，そういうことになっているらしい）であり，これだけの演出がされてしまうと，想像を絶するインパクトがあるということなのだろう。

真偽のほどはわからないが，このヌード写真集による，りえちゃんの取り分は５億円だという。１冊4,500円というバカ高い写真集であり，初版10万部が一気にさばけて一説では100万部とも。100万部売れて，印税10％で計算すると，確かに，りえちゃんの取り分は５億円近くになってしまう。いやはや，だ。

対してスカを引いてしまったのは小柳ルミ子サンであろう。元々，最初に話題をまいたときから時間がたちすぎていたところに，これだけの騒ぎが別のところでまき起こってしまったのだから。問題の小柳ルミ子サンの“ヘア露出写真集”は10月に発売されたらしいのだが，とうとうサッパリ話題にならずじまい。発売されたはずなのだが，不思議と現物は見当たらない。

このりえちゃん，富士通のパソコン「FMTOWNS」のCMキャラクターとして知られる。ヌード写真集と関係あるかどうかは不明だが，富士通のCMは観月ありさ（恥ずかしながら，何者かぼくは知らない）に交代するという。

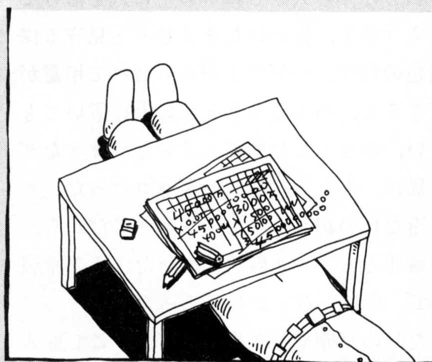
さて，ものすごい発行部数でものすごい印税を得ているといえば，ヌードのりえちゃんでさえ，全然かなわないというスケールの人物がいる。「幸福の科学」の主宰者，大川隆法さんだ。200万人の信者や会員を抱え，その人たちが合計3000万部もの本を購入したという。１冊平均1,500円として，印税率10％で計算すると，

1,500円×3000万部×10％＝45億円 (!!!)

X-OVER・NIGHT
（クロスオーバーナイト）

［第18話］

BOOK



TAKAHARA HIDEKI 高原 秀己

という破天荒な金額になってしまう。

フライデー廃刊要求騒動があり，東京ドーム大集会ありと，派手な活動が一般にまで話題をまき起こす幸福の科学。その実態たるや，ぼくの知るところではないが，この単行本作戦には脱帽する。印税の計算をしてみたが，本を発売している出版社からして「幸福の科学出版」なる傘下の企業なのだ。これほど合法的で，効果的な集金方式は，これまでの宗教ビジネスでは存在しなかったのではないか。だいたいが新興宗教の布教作戦といえば，小冊子を自前で印刷して，一般家庭のポストに無償で投函していくのが常套手段。これでは赤字が出る一方で，効果のほども期待できない。

これに対して，幸福の科学の単行本作戦は，もともと単行本にはつきものの宣伝費がそのまま宗教のPRにもなるし，販売ルートも通常の書店だから，特別な出費が必要なわけではない。ここまで来ると，幸福の科学は，“宗教ビジネス”の新たなパイオニアといったほうがいいのかもしれない。ぼくのように幸福の科学とはまったく関係がなく，本も試しにパラパラと目を通してみた程度の人間でも感心してしまう。

バブル経済が破綻したことで，土地や株，絵や金で素人が大儲けする目はなくなった。またいずれ儲かることもあるだろうが，2,3年前のようなことは，もうないだろう。パソコンやファミコンのプログラムを素人が書いて，莫大な財産が残せるような時代も，もう終わったようだ。一攫千金の儲け口がゴロゴロしている時代は，しばらくやってこないのかもしれない。

それだけに，単行本ベストセラーによる成功譚は，貴重な数少ない儲け話のひとつとなったようだ。バブル経済の破綻と同時に出版をめぐる話題が続いているのは妙に符号するような気がしてしかたがない。

とくに単行本出版で妙味があるのは，売れれば売れたで莫大な儲けとなるのだが，全然売れなくても着実な収入になるという点だろう。

かくいうぼくも，今年１冊，およそベストセラーとは縁のない単行本を出版しているが，そこそこまとまった収入にはなる。これは誰が書こうが同じことなのであるが，最初に印刷した分（初版）の数千冊がさばければ，１万部単位のラインとなり，こうなると100万円以上の収入になる。100万円という金額は，それを何回か続けて生活費をまかなうには全然足りない金額ではあるが，余剰収入として考えれば，かなりの金額である。宝くじや競馬で100万円以上取ることは難しいのだから。

これ以上の10万部，100万部ラインというベストセラーになると，もう左ウチワである。ベストセラー作家や人気コミックを別にすると，通常の本ではとても考えられない数字ではあるが，数年に１冊登場するといわれる，なぜか異常に売れる本，暴露本，芸能人関係の本などで起こる。

どうですか？　あなたも何か狙ってみませんか？



第65回

猫とコンピュータ
冷凍しちゃうぞ

Takazawa Kyoko
高沢 恭子

時間があるときに料理しておいて，食べたいときに解凍すればよい。冷凍保存って本当に便利ですよね。できれば家族そろって食卓を囲めるのがいちばんなんですけど，いまの日本って忙しい人が多いから，ね。

季節の使者，台風は，週末ばかりをねらって，つぎつぎおとずれた。9月はすべて雨の中。そして，空は灰色のまま10月を迎えた。

窓ガラスに雨がこしらえたドット模様を，わずかの晴れ間に薄日が照らす。気になってきれいにしてみたが，もう新しい雨と風がその上を叩いている。日本列島の上空は台風の定期航路のようになった。きょうも視界はぐっしょりだ。

冷たい・コワイ・痛い

ガラス戸をあけると，雨の粒と湿った風が入り込んできた。しかたなくとじようとしたとき，いっしょに外のようすをうかがっていたホンニャアの足を，すこし踏んでしまった。

雨の勢いを知ったホンニャアは，私が数秒でガラス戸をしめるのを察して，すばやく外に飛び出した。

猫のパトロールは，天候によって見合わせるということはないらしい。この鉄則に猫の体はしぜんに動きだすかのようで，雨や雪はコワくてしかたないホンニャアなのに，前後も考えずに出かけていく。

「こんな日に，たいした用事もないくせに」と思うのは人間のあさはかさで，もともと身ひとつ，道具をいっさい持たない猫にとって，いちばんたいせつなものはテリトリー，失うものはほかにないのだから，これを死守しなくてはならない。猫の社会での自分の領分を確保するためなら，雨も嵐も目に入らないのだろう。

そうでないときのホンニャアは，雨はもちろん，水もお湯も大キライ。入浴やシャンプーなんか，死んだほうがいいと思っている。

雪もコワい。ホンニャアの「武勇伝」ならぬ「イクジなし伝」の中に，「雪の日のトイレ」がある。

S市の家の庭が，一面，雪でおおわれた日のこと。軒下でソワソワ，ウロウロしているホンニャアにピンときた夫が，小さなクマデを片手に彼を抱き上げた。冷たい白い雪がコワくて，用を足すことができないのだ。夫はホンニャアにたずねながら，場所選びをした。「この辺でいいかい？」それはクルミの木の下に決まった。

自分の体の大きさに雪の中に穴をあけてもらうのを，抱かれたままじっと見守る保護色の白猫。やがて土があらわれて用意ができると，ハイどうぞとその上に置いてもらい，ホッとした顔でトイレを済ませたダメ猫は，もちろん帰りもダッコだった。

冷たいのがキライなホンニャアだから，冷蔵庫もちょっと敬遠していた。その冷蔵庫で，今回，ひどい目にあった。

ただの戸棚や引き出しなら，どこでも入ってみたくなるホンニャアだ。ハコや紙袋，段ボール，押入，せまい空間が好きだ。でもさすがに冷蔵庫だけは入らなかった。

あの冷気と特有の生臭さ。第一，中に電灯がついていて，いつもかすかにウナリ音がしている戸棚なんて，おそろしいシカケがあるにちがいない。

それでも誰かが冷蔵庫の扉をひらくちょっとした瞬間に，あの中が気になることはよくあった。食べ物が入っているようだし，どうもたいせつな戸棚らしい。

ついさそわれて，ということは誰にでもある。私が，冷蔵庫の野菜を収納する引き出しをスッと引いたとき，ホンニャアは近寄って思わずのぞきこんだ。衣類のつめかえでタンスの引き出しをあけると，めざとく見つけてやってくる，あの呼吸でつい引きよせられた感じもある。

そのとき私は，夕食に使うための肉を最上段のフリーザーから出しておくことを思いつき，野菜の引き出しをしめて立ち上がった。フリーザーの中は，わけがあってこのところかなり混雑していたために，目あての肉の包装を取り出したとき，いっしょに別のものがすべり落ちて，それが，気の毒にも，なんとホンニャアの背中に命中してしまった。

ホンニャアははじけるように飛んで，どこかへ走っていった。そのあわてぶりにはこちらも驚いたが，それから何日かあとに，同じように私の足の上に，もっと小さな冷凍肉が落ちてきたとき，ホンニャアの苦しみがよくわかった。それは石よりも硬い凶器だった。

おいしく凍らせたい

このところわが家のフリーザーをいっぱいにしているのは，スーパーから買い求めた肉や魚だけではない。私が調理したもののパックもたくさんある。

ステーキやカツ，魚の煮つけなどのたんぱく類から，きんぴらやゴマあえ，精進あげなど，野菜も多い。

数か月前までは，わが家と冷凍食品とはまったく無縁だった。食事も料理もオンライン－リアルタイム－システム。いま食べるための目的だけで調理して，その場で消化する。たとえ残っても保存する考えはなかった。

「ホームフリージング」という言葉を聞いても，調理の時間に制約のない自分が，こしらえたものをわざわざ凍らせてとっておくなんて，フリーザーと電子レンジを使ったホビーにしかならないと思っていた。冷凍と解凍の手間とエネルギーをかけて，味も鮮度も落としてから食べてみるなんて，

ほんとにご苦労さんだ。

ところが，夫が東京とS市の家を行き来するようになってから，食生活のスタイルが変則的になってきた。いつも3人そろってリアルタイムで食事を済ませることばかりではなくなってきたのだ。

いっぽう，S市の家では，どうしても夫だけで過ごすことが多い。そこでは仕事に集中するために，食事の用意をはじめ，家事のたぐいはどうしても最小限になる。そうじ，洗濯は時間を選んで集約できるけれど，食事だけはそうはいかないし，健康の基本にかかわってくる。

かりに時間があったとしても，ひとりぶんの食事の用意はむずかしいものだ。材料のムダ，味かげんのむずかしさ。おまけに1日30品目を満たそうなんて考えたら，うんざりだ。

はなれた空間にいる家族が，できるだけムダをはぶいて，内容上も好ましい食事をする方法は何か。材料と調理の手間を一括して，食事の時間だけを分離させること。その時間差を可能にしてくれるのが，フリージングだった。

いままでまったく無知だった冷凍についての実習がはじまった。こしらえた料理を手あたりしだい実験的に冷凍して，ふたたび解凍しては試食した。

おおまかにいうと，肉，魚介類などは冷凍に強い。とくに，ステーキやカツなど，油やコロモでコーティングされた状態のものは，冷凍，解凍されても味をそこなうことが少ない。野菜も，いためたものや，一度油でくるまれてから煮たもの（きんぴら，きりぼし大根，ひじきなど）は，ほとんど問題なく味が復元されるけれど，純粋な煮物となると，解凍時に水分をすっかりうばわれて，弾力のない，野菜のカスのようになってしまう。

冷凍室の使い方についても，マニュアルをはじめて手にした。保存する温度帯によって，チルド（0℃近辺で食品が凍る直前の温度）とパーシャル（－3℃近辺で食品を微凍結させる温度）があり，それぞれに適した食品がある。

また，急速冷凍の機能を使って，一気にフリージングすることが，食品のうまみや栄養分をのがさないということも知った。食品は－1℃から－5℃までの温度帯（最大氷結晶生成帯）を通過するときに，大部分が氷の結晶に変わる。この温度帯をゆっくり通過させると，氷の結晶が大きくなり，食品の組織が破壊されてしまう（東芝冷凍冷蔵庫取扱説明書による）。

こうして，東京で食事のたびに調理されるものの一部が，可能なかぎりフリージングされ，冷凍ケースでS市に運ばれる。

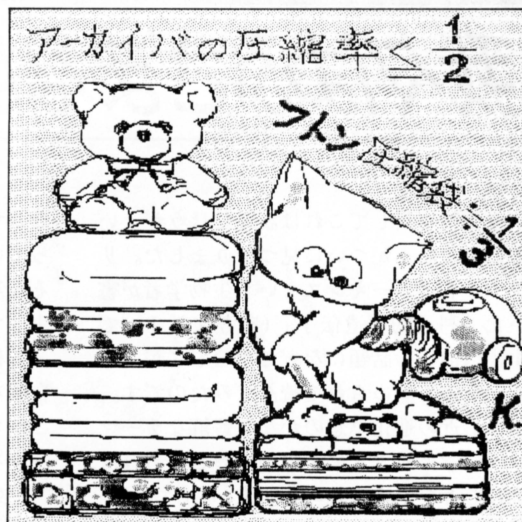


illustration Kyoko Takazawa

冷凍・解凍プログラム

凍っている食べ物を見ても，なかなか食欲はわかないもので，逆の気分にもなりかねない。慣れるまでは，出来上がりを想像しながら，思い切って解凍するという感じなので，本来の食事の楽しみとはずいぶんちがう。

S市での食事のメニューには，生野菜やみそ汁など，リアルタイムの調理も加えられるけれど，全体感としては宇宙食に近い。便利と思う半面，非常食を食べているような不満もかくせない。

でも，その中にはすでにリアルタイムの技術と時間が凍結されているのだし，それが数分で復元されるのだから，いたって合理的なはずだ。

テレビ放送のビデオ録画や，パソコン通信でのダウンロード（書き込まれた記事や情報を，自己のディスクに収めること）は，これに似ている。

リアルタイムで楽しみながら収録もできる。好きな時間にひらいてみれば，そこはオンラインの世界。チャンスがなければ無理にあけてみなくてもよい。早送りで不要の部分をカットするワザは，収録したものでなければありえない。時間の圧縮と解凍が思うままだ。

じっさいにパソコン通信では，送受信の時間短縮をはかるために，「冷凍」「解凍」のプログラムが使われている。とくに，オンラインソフトウェアのアップロード（ネット上への登録），ダウンロードのためのプログラムだ。

オンラインソフトウェアはネット内に登録されている，おおやけに提供されたプログラムで，通信のための必需品ともいえるプログラムをはじめ，貴重な作品がたくさんある。著作権のうえでフリーソフトウェア（利用については無料だが，作者に著作権がある）と，シェアウェア（長く利用する場合は作者に一定額を支払う）に分かれるが，かたちのうえでは自由にダウンロードできるようになっている。

これらの公開されたプログラムは，ほとんどが実行ファイルやマニュアルなど，いくつかのファイルをひとまとめにして圧縮（アーカイブ）したものなので，解凍しなくては利用できない。

自己解凍型のプログラムは，実行することで解凍されるけれど，そうでないものはそれぞれの圧縮形式に従ったコマンドで，解凍をほどこす。冷，解凍のための圧縮プログラム（アーカイバ）は，同じくパソコンネット上で入手できる。

ホンニャアは，冷蔵庫にはとうぶん近寄らないだろう。名高いミステリーの短編に，若妻が冷凍の羊肉で夫の頭を強打し，殺害するのがあった。読んだ当時は，冷凍肉を利用する生活習慣がいかにもエキゾチックに想像され，その重さや硬さに実感がなかったので，メルヘンのように思った。でも自分の足に痛打をあびてみたら，とてもおかしくてこわい話に変わった。

待ちわびていた夫が帰宅して会話するうち，思いがけないことから動転して，殺してしまう。凶器になったのは夫のために用意しておいたおいしい肉だった。身につまされる発作的な犯行だけれど，警察官の夫の同僚が捜査にやってきて，凶器が見つからないまま夜がふけると，解凍，加熱された羊の肉が彼らにふるまわれる。

凍った肉が熱くなる落差がすごい。

我々はぶざまな巨大ロボット？

全生物に対してこれは読んだほうがよいと勧めたい本がついに見つかりました。リチャード・ドーキンスという生物学者が書いた「利己的な遺伝子」(参考文献1）です。以前から話題になってたのですが，知人から強く勧められて読んでみたのです。

「最近の進化思想のかなり難解なテーマが，科学に対する最小限の素養さえあれば，誰にでもたやすく読める」(米「サイエンス」誌）ように書かれており，しかも「この重要な本はこれ以上ないほど面白い」（米「エコノミスト」誌）ので，あっという間に読み上げてしまいました。

さすがに世界中で大きな反響を呼び起こしただけのことはある本です。読後のなんとも形容のしがたい感じは，景山民夫氏が初めて大川隆法氏の書物を読んだときに受けた衝撃や感動にひけをとらないと確信します。そこで，今月はぜひぜひこの書物を読んでほしいという一心で書きました。

解かれるなぞなぞ

本書は動物や人間社会で見られる種々の社会的行動が，なぜこのように進化してきたかを解き明かそうとするものです。特徴的なのは，生物の行動の裏にある原理を，各個体や個体群の中にではなく，自らのコピーを増やそうとする遺伝子の利己性に見出そうとしているところです。そして，通説となっている「種の保存」のような概念を，実体のないものだとして攻撃します。

この本の真価は，ゲーム理論を応用することにより，ダーウィニズムというものが遺伝子のレベルできわめて説得力を持つというところにあると思われます。しかし，同時に人々の現実の生活様式や道徳観念に結びつけやすいという事実もあります。本書の中ではそのような傾向をなるべく排除しようとしていますが，理解の助けになる場合がありますから，そういうものも取り上げることにします。

遺伝子のどこが利己的？

この本では種々の例を通して遺伝子と進化の謎を解き明かしています。説いていることの核心をひと言でいえば，「すべての生物は自己複製を行う実体（遺伝子）の生存率の差に基づいて進化する」ということになるでしょう。なにかむずかしいことをいっているように思われるかもしれませんがそんなことはありません。たとえば，「現在生きている我々の先祖は全員早死にもせず，ある程度以上元気だった」ということなら，自明なことであると納得いくでしょう。ここから発展させて，我々の性質や行動パターンなどに対して多かれ少なかれ子孫に影響を残すという性質を持つ「遺伝子」というものに着目すればよいのです。

ダーウィニズムの適者生存などの原理にも近いのですが，もっと具体的です。生存率の低くなってしまうような性質や，行動パターンをそれが宿る生物個体に示させる遺伝子は，統計的にはどんどん少なくなってしまうでしょう。なぜならば，生殖する前に生物が死んでしまう確率が高いので，時間の流れにしたがって減っていくことが予想されるからです。逆に生存する確率の高い性質を示す遺伝子はあとあとまで生き残っていくことでしょう。

ここで遺伝子が突然変異するという現象を考えに入れる必要があります。そうしないと遺伝子は進化せずに減少していくだけになりますから。突然変異で偶然生存率の高まった遺伝子が現在まで生き残ってきたのです。もちろん進歩発展だけだったわけではありません。多くの突然変異は生存率を低めたが，数億年かけた結果，生存率の高まる遺伝子が蓄積されたのでしょう。

もう一度基本的なところを繰り返しますと，自分の数を減らさず増やすような性質をそれが宿っている生物個体に示させるような遺伝子が優位になっていき，時間の経過につれてそうでないような遺伝子を絶滅に追い込んでいくというのです。

生命を作る

本書は動物や人間の社会的行動を遺伝子の進化という観点だけで読み解こうとするものであり，地球上における生命誕生の過程は必ずしも本書の中心的なテーマというわけではありません。しかし，生命誕生についても，まったく同様の明快な推測が述べられています。注目すべきは，分子の初歩的な進化が物理や化学の普通のプロセスで起こりうるということです。このことは，生命誕生以前の地球の化学的状態をまねた室内実験で，遺伝物質DNAの構成要素が作り出された証拠とも述べられています。

重要なのは，数億年の間に自分を複製する（親和性の高い分子とペアになるような性質）能力を持った自己複製子が誕生したことと，コピーの間違いを繰り返すうちに，長生きで，多産性で，コピーが正確な分子の含有率が高くなっていったということです。さらに，そのような分子間で競争相手を排除するような性質をたまたま持った物質が残っていったのです。しだいに，身の周りに蛋白質の物理的な壁を持つものが現われたのです。これが細胞です。さらに自己複製子のうち自分の住む生存機械を築くものが出現しはじめたのです。それが我々の体です。彼らの維持こそ我々が存在する最終的な理由なのです。彼らには遺伝子という名前が付けられているのです。

一見自己犠牲に見えるが……

なぜ動物が一見利他的に見える行動をするかという問題は，ダーウィニズム理論にとって難問であると考えられてきましたが，この問題に対しても明快な答えが記述されています。鳥の警戒音を例にとって説明しています。集団で生活している鳥の群れをタカが襲ったとき，最初に発見した鳥が警戒音を発し，自分が犠牲になってもタカの攻撃から仲間を守ろうとする行動です。

いくつかの考え方が紹介されていますが，いちばん明快なのはその群れが近親の血縁を含んでいるときです。群れに近親が多く含まれているときに警戒音を発することを促すような遺伝子は，生存確率が高いであろうと予測されるからです。つまり，たとえこの警戒音で自分（ここではその生物個体）が死んでしまったとしても，それで救われる近親者が多ければ，結果として同一の遺伝子が多く残っていくからです。

オスとメスの数が等しい理由

ほとんどの動物でオスとメスの数が等しくなっている理由についても具体的な説明がなされています。きわめて単純化すると次のようになります。もし，オスがメスよりぐんと少なくなると，オスに生まれることは生殖して子孫を残せるチャンスが多くなることを意味するので有利になります。

したがって，オスばかり生むような傾向を持たせる遺伝子の数が世代交代につれて増えていき，結果的にオスばかりになっていきます。今度はメスばかり生むような遺伝子が有利になり……，ということが繰り返されるので，どちらかに偏るということはないというわけです。実際にはこのような増減を繰り返さず，一定の値を取ります。

人間の寿命を数百歳にする

遺伝子の中には，生まれて何年かたったら，その生物個体を（たとえばガンで）殺してしまおうとするような遺伝子（致死遺伝子）があります。生まれたばかりですぐに働こうとする遺伝子もいれば，80年ぐらいたったら働き出す遺伝子もいます。これを利用すると寿命を延ばすことが本当にできるのです。低年齢での出産を制限すればよいのです。といっても，20歳以下はだめとかいう常識的な話ではなく，40歳，50歳，60歳という非現実的な話です。こんなことがなぜ寿命を延ばすのでしょう？

子どもを作る前に死んでしまった場合，その個体の遺伝子は当然絶滅します。したがって，早い年齢で死に到らせるような遺伝子は比較的少なくなっているはずです。この場合，「早い年齢」という基準が生殖を平均何歳で行うかということに直接関わります。したがって，その基準となる年齢を上げることにより，早い年齢で働く致死遺伝子の割合を減らすことができるというわけなのです。もちろん，すき好んでこんなことをやるはずはないのですけれどね。

父親が育児に熱心でない理由

多くの生物において，父親より母親のほうが育児に熱心のようです。これには社会的，肉体的原因も当然あると思いますが，ひとつの原因として，一般に母親のほうが自分の子を確認しやすいという客観的事実があります。確かに他人の子でも熱心に面倒を見ているような遺伝子はとっくに勢力を減らしていることでしょう。人間でも子どもが自分に似ているとうれしいのは，実は遺伝子の喜びなのかもしれません。

動物たちが紳士的である理由

動物の戦いが実に紳士的であるということを説明する理論こそが，ドーキンスの本の主張するいちばん面白く，価値のあるところといえます（これは実は別の科学者の提唱に基づく部分ではありますが)。この理論は数学（ゲーム理論の応用）そのものであり，本当はそう簡単ではありませんが，本書は実にわかりやすく述べてあります。

生物が生きていくには，場面場面に応じてどういう行動を取るかという戦略が必要になります（その戦略も遺伝子が左右します)。ここでゲーム理論が登場します。ただ，その場合，ある戦略を取ってそれで1回うまくいけばよいというのではなく，群れの中で世代交代を経てバランスが取れて安定な戦略が重要になります。これは進化的に安定な戦略（evolutionarily stable strategy）と呼ばれます。

動物間の関係も平和的なものと攻撃的なものに大きく分けられます。攻撃的になれば，それだけ自分に対するリスクも大きくなります。これをゲーム理論で定量的に評価すば，現在のような紳士的な態度を取るような遺伝子の優位性を理由づけることができるというのです。

遺伝子万能主義に対する疑問

僕自身，この本の基本的な部分に関しては異存はないのですが，やはり，完全にランダムな遺伝子の突然変異だけでここまで進化しうるのかという疑問はあります。しかし，それははっきりとした疑問というわけではありません。なぜなら，数億年という想像を絶する時間に対するイメージがまるでわかないからです。ミクロなレベルの量子理論が納得しづらいのと同じです。

しかし，もしこれが完全なランダムな突然変異で説明できないというのならば，別の原理も必要となってくるでしょう。そのような原理は，たとえば，次のような要素を含むものかもしれません。

1)　全体と部分の問題

完全にミクロな部分を記述するだけでは，本当に問題を解き明かしたことにはならないのではないかもしれないということです。部分と部分，あるいは部分と全体の間の相互関係に関する考察が必要でしょう。

たとえば，最近，人工知能研究での流行りの概念に「リフレクション」というものがありますが，これなどは生命体という全体から個々の遺伝子への影響を考えるという点で関連性があるのかもしれません。

2)　各遺伝子の持つ戦略の非固定化

この本では遺伝子の持つ形質を固定なものとしていますが，これを動的にすると生物の適応能力や進化の説明がもっとうまくできるようになるかもしれません。これは必ずしも個々の遺伝子が動的に性質を変えているというのではありません。遺伝子のまとまりとしての形質が変化する，あるいは遺伝子のまとまりの組み替えによって形質が連続的に変化するのではないかということです。1)とも関連しているでしょう。

人間の利他性は幻か？

この本での主張にはいくつかの誤解されやすい点があります。たとえば，この話が，いわゆる「遺伝か環境か論争」に関係するのではないかとか，人の生き方に対する道徳のようなものについて書かれているのではないか，などということです。

本の冒頭でこれらのことは否定されています。確かに遺伝というものがゼロでないかぎり，どのような程度でありうるとしても，すべて成り立つ話が書かれていますし，道徳に関係するような話は一切出てきません。話は脱線しますが，遺伝か環境か論争は，僕自身は不毛なものだと思っています。次のような理由からです。

1)　個人差が大きい
2)　科学的に証明することがむずかしい
3)　もし遺伝が優勢要因だと判定されると多くの弊害（悪用）が生じがちである

一方，道徳の話は案外面白い話かもしれません。利己的な遺伝子に左右されない利他的な行為が可能である人類はいったいなぜ出現したのか。なぜ，このように進化したのか。人間の利他性というのは幻なのか。実際，「対談：ヒトは立ったサルである」(参考文献2）によれば，人間とチンパンジーの遺伝子の差は1,2%であり，ウマとシマウマぐらいの差しかないというのです。

参考文献

1)　リチャード・ドーキンス，『利己的な遺伝子』，(科学選書9)，紀伊国屋書店，1991.
2)　熊野純彦，広松渉，松沢哲郎，『対談：ヒトは立ったサルである』，青土社，現代思想1991年10月号.

Oh!X INDEX'91

特集

急接近！ SX-WINDOW

- 美しい環境を目指して……1, 64
- SX姫と15人の小人たち……1, 66
- ウィンドウプログラミングへの道……1, 70
- ライフゲームSXLIFE……1, 87

グラフィックの"実験的"手法

- CGの基礎……2, 33
- ドロー系グラフィックツールの魅力……2, 36
- ポリゴンデータ「3D倶楽部」……2, 42
- Z'sSTAFF支援ツールZ's-EX……2, 43
- 半影つきレイトレーシングHASH.X……2, 50
- 製品試用レポートFine Scanner-X68……2, 56

SX-WINDOWプログラミング

- C言語によるプログラミング……2, 100
- コラム 目玉を小さくするプログラム!?……2, 106
- GRAPHMANを使ってみよう……2, 107
- ポップアップメニューの追加……2, 116
- コラム「SXエンターテイメントキット」計画……2, 120

MIDI & MUSIC PROCESSING

- コンピュータミュージック入門……3, 34
- ミュージックツール実践活用……3, 37
- Musicstudio PRO-68K ver.2.0……3, 40
- MUSIC1.FNCで遊ぶ……3, 42
- ヴァルナより町のテーマ……3, 47

人とゲームのインタフェイス

- 戦うインタフェイス……4, 46
- 資源の賢い活用を……4, 48
- 模擬飛行と鍵盤……4, 52
- マウスで操る自動車レースIndianapolis500……4, 56
- 人に優しいって, なんだ？……4, 57
- ゲーム空間へのインタフェイス……4, 64
- アドベンチャーゲームを救え……4, 67
- ゲームをつくる「質感」とは……4, 70

X68000 XVIとSX-WINDOW ver.1.10

- X68000 XVIの製品概要……5, 58
- 速報SX-WINDOW ver.1.10……5, 61
- 速くなったFLOAT?.X……5, 66
- 誕生からXVIへ……5, 68

初心者のための環境構成術

- まずはハードウェア環境の整備から……6, 44
- 三種の神器 DIR/CD/TYPE……6, 46
- COMMANDマスターへの道……6, 50
- SX-WINDOWで環境をつくること……6, 57
- 基本はテキストファイル……6, 59

Personal Tool,BASIC

- X-BASICの基礎……7, 74
- カットファイルを作成しよう……7, 77
- MMLを画面に表示する……7, 80
- スプライトを加工する……7, 85
- X-BASICでMAGICを……7, 89

印刷の世界へ

- 出力デバイスを探る……8, 82
- 基礎からのカラー印刷……8, 88
- HighFidelityへの挑戦……8, 95
- PostScriptとはなにか……8, 97
- TeXからのアプローチ……8, 99
- IOCS用FONT200書体……8, 105

Brush up your MAGIC.

- MAGICの原理と活用……9, 50
- グラフィックパッケージMAGIC ver.2.0……9, 52
- 3D関数の基本操作……9, 72
- MAGIC.FNCとMAGICダンプリストの入力法……9, 77

マシン語との邂逅

- マシン語の考え方……10, 74
- IOCS,DOSコールの考え方……10, 77
- デバッガにて理解されたし……10, 82
- 避けて通れぬ道, アドレッシング……10, 90
- 実践アセンブラプログラミング……10, 94

空間彷徨型ゲーム大分析

- パワーモンガー……11, 74
- スターウォーズ……11, 80
- ドラッケン……11, 84
- アーケードゲームにおける3D体験……11, 89
- 立体空間の料理法……11, 93

音・そして音楽とコンピュータ

- 音とはなにか……12, 74
- 冬の夜長のスペクトル解析……12, 76
- FM音源の波形を創る……12, 81
- Z-MUSIC公式ガイドブック……12, 90
- Z-MUSIC LIVE SHINDO ON STAGE……12, 93
- MIDIをめぐる環境……12, 99
- MIDI出力方法論……12, 101

特別企画

第6回 言わせてくれなくちゃだワ

- カラーイラスト大集合 Oh!X readers'ぎゃらりぃ……5, 36
- micro Communication 言わせてくれなくちゃだワ……5, 89
- ざ・質問箱SPECIAL……5, 106

創刊9周年記念

- PC-9801用マウスを使う……6, 40
- マルチウィンドウシステム System-7C……6, 88
- 愛読者特大プレゼント/モニタ……6, 96

特別レポート

- Oh!Xの正しい読み方……9, 81
- GSフォーマットを斬る……10, 104
- X68000ゲームソフトのゆくえ……10, 148

付録ディスク

謹賀新年PRO-68K

- 付録ディスク利用の手引き……1, 98
- VS2.Xの使い方……1, 100
- システムおよびツール類……1, 102
- CARD2.FNC & CARDDRV.X……1, 106
- Z'sSTAFF支援ツールZ's-EX……1, 112
- グラフ作成ツールMichael……1, 117
- SX-WINDOW開発セット＆アクセサリプログラム……1, 120
- ウイルス検出プログラムDOCTOR2.X……1, 123

黄金週間PRO-68K

- カラー紹介 黄金週間PRO-68K……5, 38
- 付録ディスクの内容と使い方……5, 72
- 表計算ソフト tinyCalc……5, 75
- グラフィックツール CARDSHOP……5, 79
- グラフィックパッケージ MAGIC……5, 81
- 新・魔術師への道……5, 86
- 表計算ソフト tinyCalc……6, 109
- MAGICの拡張……6, 113
- Digital Arajin & SXWHERE……6, 116
- SXIMAGE.X……6, 118
- tinyCalcの関数を作る……7, 100

THE SOFTOUCH

THE SOFTOUCH SPECIAL

- 1990年度 GAME OF THE YEAR ノミネート作品発表……2, 26
- 決定！1990年度 GAME OF THE YEAR……4, 36
- またまた勝手にGAME OF THE YEAR……4, 41

話題のソフトウェア

- プリンス・オブ・ペルシャ/パロディウスだ！/エメラルドドラゴン/マーブルマッドネス/ザ・スーパーラスベガスII/アステロイド・クイーン/JANJON/デルタアーム/ニニンバトル/生中継68/ラプラスの魔/中華大仙/栄冠は君に……1, 37
- ノスタルジア/リングマスターII/マジカルショット/ザークレジェンドスペシャル/スライス/大航海時代/スペースローグ/ブルトン・レイ シナリオエディタ/レインフォーサー/アトミック・ロボキッド/ファランクス……2, 81
- メルヘンメイズ/シグナトリー/ファランクス/ボンバーマン/中華大仙/マーブルマッドネス/アルガーナ/Misty vol.7/サブナック/電脳倶楽部……3, 81
- スコルピウス/サブナック/eXOn/シムシティー テレインエディター/マーブルマッドネス/ノスタルジア/マジカルショット/クォータースタッフ/Ko-WINDOW/COMET……4, 73
- ロードス島戦記/遙かなるオーガスタ/キャンペーン版大戦略II/サイレントメビウス/ノスタルジア/スコルピウス/マーキュリー/マジカルショット/ぴんぴん麻雀ピーチエンジェルデータ集/シューティング68K……5, 43
- 黄金の羅針盤/ドラッケン/プリンス・オブ・ペルシャ/大戦略III'90/ダッシュ野郎/スターモビール/マーキュリー/ロードス島戦記～灰色の魔女～……6, 27
- 生中継68/キャメルトライ/装甲騎兵ボトムズ DEAD ASH/アクアレス/オルテウスII/Easypaint SX-68K……7, 30
- イース/3D 2(仮称)/F15ストライクイーグルII/生中継68/アクアレス/ライヒスリッター/ループス/インペリアルフォース/ドラゴンウォーズ……8, 27
- ボナンザブラザーズ/3D 2(仮称)/飛翔鮫/銀河英雄伝説IIDX+kit/シュバルツシルトII/ロードス島戦記～灰色の魔女～/ジーザスII/グループ・エックス/スーパー上海ドラゴンズアイ……9, 29
- パワーモンガー/ドラッケン/機動戦士ガンダム クラシック・オペレーション/ゼノン2/麻雀マスター/ノア/コラムス/スターウォーズ/TAKERU関連諸々/ストリートファイターII大会観戦記……10, 26
- 出たな!! ツインビー/フェアリーランドストーリー/プロサッカー68/ディノランド/アルシャーク/ノーブルマインド/ダーウィンズジレンマ/麻雀マスター/HEAVY NOVAゲーム大会……11, 33
- 大戦略III'90/ブリッツクリーク/ラストバタリオン/NIKO2/PITAPAT/ヴェルスナーク戦乱/プロサッカー68大会……12, 32

GAME REVIEW

- ソル・フィース……1, 40
- 銀河英雄伝説II……1, 42
- シュヴァルツシルト……1, 44
- 続ダンジョン・マスター カオスの逆襲……1, 46
- ワールドスタジアム……1, 48
- ハイドライド3SV……1, 50
- ブルトン・レイ……1, 52
- 栄冠は君に……2, 84
- KLAX……2, 86
- ダイナマイト・デューク……2, 88
- エメラルドドラゴン……2, 90
- プール・オブ・レイディアンス……2, 92
- アトミックロボキッド……3, 84
- スペースローグ……3, 86
- ラプラスの魔……3, 88
- 続ダンジョン・マスター カオスの逆襲……3, 90
- メルヘンメイズ……4, 76
- 中華大仙……4, 78

スライス……4, 80
ボンバーマン……4, 81
哭きの竜……4, 82
リングマスターII……4, 83
マーブル・マッドネス……5, 46
シグナトリー……5, 48
石道……5, 50
クォータースタッフ……5, 52
サブナック……5, 54
パロディウスだ！……6, 30
遙かなるオーガスタ……6, 32
ノスタルジア……6, 34
マジカルショット……6, 36
ファランクス……7, 32
スコルピウス……7, 34
A列車で行こうIII……7, 36
キャンペーン版大戦略II……7, 38
パロディウスだ！……7, 40
マーキュリー……7, 42
シューティング68K……7, 43
シムシティー テレインエディター……7, 46
黄金の羅針盤……8, 30
サイレントメビウス……8, 32
パロディウスだ！……8, 34
装甲騎兵ボトムズ DEAD ASH……8, 36
ダッシュ野郎……8, 37
エイトレイクス ゴルフクラブ……8, 38
AIIIマップコレクション……8, 39
イース……9, 32
生中継68……9, 34
アークス・オデッセイ……9, 36
信長の野望・武将風雲録……9, 38
ループス……9, 40
スターモビール……9, 41
ドラゴンウォーズ……9, 42
エイトレイクス ゴルフクラブ……9, 43
ボナンザブラザーズ……10, 30
ロードス島戦記～灰色の魔女～……10, 32
ジーザスII……10, 34
シュヴァルツシルトII 帝国ノ背信……10, 36
銀河英雄伝説IIDX＋……10, 38
インペリアルフォース……10, 39
生中継68……10, 40
キャメルトライ……11, 36
アクアレス……11, 38
フューチャーウォーズ……11, 40
オルテウスII……11, 41
フェアリーランドストーリー……12, 34
プロサッカー68……12, 36
機動戦士ガンダム クラシック・オペレーション……12, 38
ノーブルマインド……12, 40
サイバーコア……12, 41
F15ストライクイーグルII……12, 42

AFTER REVIEW

シムシティー……1, 54
ラグーン……2, 94
闇の血族……3, 92
遊撃王II(エアー・コンバット)/GUNSHIP……5, 56
ソルフィース/ナイアス……6, 38
プリンス・オブ・ペルシャ……8, 40
メルヘンメイズ……9, 44
ファランクス……10, 42
遙かなるオーガスタ……11, 42
ボンバーマン……12, 44

全機種共通システム

ブロックアクションゲーム COLUMNS……1, 160
ダイスゲーム KISMET……2, 138
アクションゲーム MUD BALLIN'……3, 128
SLANG用カードゲーム DOBON……4, 130
実数型コンパイラ言語 REAL……5, 148
Small-C処理系の移植……6, 138
実数型コンパイラ言語 REAL ソースリスト編……7, 122
Small-Cライブラリの移植……8, 142
SLANG用NEWファイル入出力ライブラリ……9, 152
Small-C活用講座(初級編)……10, 142
Small-C活用講座(応用編)……11, 122
アクションゲーム MORTAL……11, 126
Small-C用SLANGコンパチ関数……12, 142

連載・シリーズ

知能機械概論

第44回 ジョブズはやっぱり天才だ！……1, 172
第45回 感涙もののマシン語プログラム……2, 154
第46回 いまこそエコロジカルなハイパー進化論を！ 3, 154
第47回 復刻版面白玉手箱……4, 162
第48回 できることはすぐにしなさい型……5, 142
第49回 肥大したアザラシの群れの中で……6, 174
第50回 急にNeXTを使いはじめる……7, 155
第51回 ポンコツ計算機を売る法，あるいは今世紀最後の教科書……8, 155
第52回 キーワードは貧乏……10, 158
第53回 電子の海に沈むアンディ・ウォーホル……11, 155
第54回 我々はぶざまな巨大ロボット？……12, 156

猫とコンピュータ

第55回 過激なCRTと共に……1, 174
第56回 楽しめるRPGギフト……2, 156
第57回 青春コミケ＆パソケ……3, 156
第58回 脳ミソをくれえ……4, 164
第59回 ファジィの親分……5, 144
第60回 TK-復活の午後……7, 158
第61回 FAX見つけた！……8, 158
第62回 まだまだCOLUMNS……9, 162
第63回 コロッとディスクカバー……10, 156
第64回 夏の日のメンテナンス……11, 158
第65回 冷凍しちゃうぞ……12, 154

X-OVER NIGHT

第8回 パソコン戦線異状なし……1, 171
第9回 街の空気……2, 153
第10回 ニュース欠乏症候群……3, 153
第11回 アメリカ人の気質……4, 161
第12回 ハイテクも使いよう……6, 173
第13回 内側から見たアメリカ……7, 154
第14回 ビデオ時代の転換期……8, 154
第15回 情報収集の妙味……9, 160
第16回 買い換え……10, 153
第17回 ムダむだ無駄……11, 154
第18回 BOOK……12, 153

大人のためのX68000

第4回 電話番号が変わります……1, 129
第5回 FIXER ver.4.0……2, 96
第6回 発売間近！ CARD PRO-68K ver.2.0……3, 98
第7回 プリンタで紙資源浪費のこと……5, 137
第8回 第2回愛読者アンケート結果大分析大会……6, 99
第9回 CARD PRO-68K ver.2.0の基礎……6, 104
第10回 CARD PRO-68K ver.2.0の応用……7, 54
第11回 画像処理と称して遊ぶ……8, 45
第12回 Multiwordは救世主となるか……9, 97
第13回 脳の欲望が指先を動かす……10, 49
第14回 Multiwordの救世主となるか……11, 46
第15回 F CARD-GTをいじくる……12, 53

X68000マシン語プログラミング

Chapter_14_H ソーティングプログラム(前編)……2, 69
Chapter_15_H ソーティングプログラム(後編)……3, 65
Chapter_16_H 必須のラインルーチン(その1)……5, 121
Chapter_17_H 必須のラインルーチン(その2)……6, 69
Chapter_18_H 多角形の塗り潰し……7, 65
Chapter_19_H グラフィックパターンの拡大・縮小……8, 117
Chapter_$1A_H$ グラフィックパターンの回転……9, 129
Chapter_$1B_H$ シードフィルによる塗り潰し……10, 119
Chapter_$1C_H$ 円描画ルーチンの作成……11, 103
Chapter_$1D_H$ 自由変形ルーチンの作成……12, 111

ようこそここへC言語

第4回 配列って何だろう(その1)……2, 64
第5回 配列って何だろう(その2)……3, 56
第6回 文字列って何だろう……4, 122
第7回 関数って何だろう(その1)……5, 129
第8回 関数って何だろう(その2)……6, 73
第9回 式と演算子って何だろう……7, 110
第10回 標準入出力って何だろう……8, 108
第11回 ポインタって何だろう(前編)……9, 119
第12回 ポインタって何だろう(後編)……10, 109
第13回 構造体って何だろう……11, 109
第14回 ファイル入出力って何だろう……12, 119

マシン語カクテルin Z80's Bar

第18回 乱数は世界を救うか……2, 130
第19回 限りある資源をハフマンで……3, 146
第20回 事故の前にブレーキング……4, 150
第21回 これで完成？……5, 117
第22回 最後の手段を……6, 129
第23回 アフターケアなの？……7, 138
第24回 もうどうにも止まらない……9, 137
第25回 秋の運動会スペシャル……10, 131
第26回 怒りのデバッグ……11, 143
第27回 炎のプログラミング勝負……12, 133

(で)のショートプロぱーてぃ

その16 クリスマスにデモを……1, 56
その17 行け行けユーティリティ……2, 146
その18 春のピコピコ！……3, 120
その19 育てや育て，プログラム……4, 145
その20 この木，踊る木……5, 140
その21 みんなで狙い撃ち！……6, 170
その22 おお，グラフぃっく……7, 142
その23 放物線も再帰も算数……8, 130
その24 小さく小さく，奥へ奥へ……9, 147
その25 ノリは駄菓子だ！……10, 127
その26 人生設計の季節……11, 98
その27 ★よもう一度！……12, 149

Oh!X LIVE in '91

めぞん一刻より 暁に鐘は鳴る/響子の悲しみ/夜の雨(X68000)……1, 124
涙で綴るパパへの手紙(X1/turbo)……1, 124
Misty Blueより オープニングテーマ(X68000)……2, 77
スプーンおばさんより リンゴの森の子猫たち(X1/turbo)……2, 77
戦いの兜(X68000)……3, 49
LITTLE WING(X1/turbo)……3, 49
リゾ・ラバ(X68000＋MIDI)……3, 49
花(X68000＋MIDI)……3, 49
Easy Come,Easy GO！(X68000)……4, 141
シシリエンヌ(X1/turbo)……4, 141
ブービーキッズより ブービー城下町(X68000)……5, 109
NO.NEW YORK(X1/turbo)……5, 109
暴れん坊将軍より 夜明け(X68000)……6, 152
不思議の海のナディアより ブルーウォーター(X68000)……6, 152
POWER HOLL(X68000)……6, 152
魔法の妖精ペルシャより 見知らぬ国のトリッパー(X1/turbo)……6, 152
今すぐKISS ME(X68000)……7, 146
歩いていこう(X68000)……7, 146
パワードリフトより SIDE STREET(X68000)……8, 63
ワンダラーズ・フロム・イースより Be Carefull(X68000)……8, 63
TURBO OUTRUNより Checker Flag……8, 63

パワードリフトより Artistic Traps(X68000+MIDI) …8, 63
One(X68000) ……9, 85
WHITE MANE(X1) ……9, 85
うれしい! たのしい! 大好き!(X68000) ……10, 135
SPANISH BLUE(X1/turbo) ……10, 135
オーダインより エンディング&コンティニュー/ROUND X ……11, 66
OH YEAH!(X68000) ……12, 58
サイレント・イブ(X1/turbo) ……12, 59
ジングルベル(X68000) ……12, 61

DōGA・CGアニメーション講座
<15>宇宙要塞CADの逆襲 その2 ……1, 132
<16>私の作品制作……3, 114
<17>CGAコンテスト座談会 ……4, 84
<18>戦えロボット君2(前編)……6, 158
<19>戦えロボット君2(中編)……9, 102
<20>戦えロボット君2(感動の完結編)……11, 60

ハードウェア工作入門
<7>センサー回路 その1 ……1, 60
<8>センサー回路 その2 ……2, 143
<9>センサー回路 その3 ……3, 124
<10>センサー回路 その4 ……4, 137
<11>メカトロニクス制御 その1 ……5, 113
<12>メカトロニクス制御 その2 ……6, 165
<13>メカトロニクス制御 その3 ……7, 140
<14>メカトロニクス制御 その4 ……8, 126
<15>ハイテクタンク製作(理論編)……9, 109
<16>ハイテクタンク製作(実習編)……10, 54
<17>ハイテクタンク製作(応用編)……11, 50
<18>ハイテクタンク製作(発展編)……12, 50

吾輩はX68000である
第1回 まずAより表示せよ ……4, 92
第2回 いでよ, 文字たち! ……6, 63
第3回 我が好敵手C言語 ……7, 59
第4回 魔法の函の正体は? ……8, 57
第5回 最後の砦, 配列と構造体 ……9, 113
第6回 グラフィックモードあれこれ ……10, 57
第7回 無敵の簡易描画法 ……11, 148
第8回 愛のIOCSコール ……12, 68

よいこのSX-WINDOW講座
第1回 制御ボタンを使う ……4, 97
第2回 ダイアログで対話する(前編) ……6, 121
第3回 ダイアログで対話する(後編) ……7, 103
第4回 アイコンのドラッグとアイコン化 ……8, 50
第5回 マウスカーソルを変更する ……10, 61

シミュレーションプログラミング入門
第2回 シミュレーションの道も絵描きから ……1, 153
第3回 シミュレーションは未来をひらく ……2, 58
第4回 冬だから, シミュレーション ……3, 75
第5回 温泉とコンピュータのファジィな関係 ……4, 107
最終回 「株式」学問のススメ ……9, 91

X68000CARDDRV用カードゲーム
EIGHT ……3, 139
スロットポーカー……2, 121
コンパイラ対応カードゲーム変更点……3, 143
THE SuperKABU……4, 88
Christmas Tree ……6, 107
七並べ……8, 42
ギャップ……11, 57

Computer Music入門
(1)まず音階, そして和音の基礎……10, 98
(2)和音の種類と構造徹底マスター……11, 53
(3)メロディを生かす伴奏とは……12, 62

清水和人流プログラミング道場
その3 段階的に鍛えていくべし ……1, 138
その4 ピアニスト不屈のエディタ ……4, 155

PASCALプログラミングへの招待
<6>実行時チェック・Cとのリンク……1, 144
<最終回>Garbage Collection in PurePASCAL ……3, 62

X-BASICプログラミング調理実習
(最終回)カード型データベース(3)……1, 148

X1/turbo用ディスク管理プログラムINTEGRAL X1
バグレポートとファイル関数……2, 124

響子 in CGわ〜るど
第1回 3次元CGってなあに? ……6, 82
第2回 進化しつづける画材 ……7, 28
第3回 脳のなかのイメージを視覚化する ……8, 74
第4回 止まる時間, 流れる時間 ……9, 26
第5回 いろいろな形 ……10, 24
ANOTHER CG WORLD ……10, 154
第6回 色のハーモニー ……11, 30
ANOTHER CG WORLD ……11, 96
第7回 プレゼント ……12, 30
ANOTHER CG WORLD ……12, 66

機種別活用・プログラム

MZ-700
マルチウィンドウシステム SYSTEM-7C ……6, 89
詳説 System-7C……7, 47

MZ-2500
ビンゴゲーム REACH(ショート) ……11, 99

X1/turbo
アルファベットの逆襲(ショート)……3, 120
音楽演奏 AUTO ORGAN.BAS(ショート) ……5, 141
アクションゲーム LASER(ショート) ……6, 171
ゲーム The Master of Payment ……7, 178
シューティングゲーム DEFEAT2 ……8, 135
アスレチックアクションゲーム Manual Runner ……9, 142
アクションゲーム MERVEL.BAS(ショート)……9, 149
アクションゲーム Z80'sBar ……10, 131
シミュレーションプログラミング入門……(→連載)
X1/turbo用ディスク管理プログラムINTEGRAL X1…(→連載)
Oh!X Live in '91 ……(→連載)

X1turbo
数式グラフ化 ぐらふくん(ショート) ……7, 144

X68000
美しい環境を目指して(特集)……1, 64
SX姫と15人の小人たち(特集) ……1, 66
ウィンドウプログラミングへの道(特集)……1, 70
ライフゲームSXLIFE(特集) ……1, 87
外字ユーティリティ USK2WP.BAS/WP2USK.BAS(ショート) ……1, 57
デモ COMET.BAS(ショート) ……1, 58
ドロー系グラフィックツールの魅力(特集)……2, 36
ポリゴンデータ「3D倶楽部」(特集) ……2, 42
Z'sSTAFF支援ツールZ's-EX(特集) ……2, 43
半影つきレイトレーシングHASH.X(特集) ……2, 50
製品試用レポートFine Scanner-X68(特集) ……2, 56
C言語によるプログラミング(特集)……2, 100
コラム 目玉を小さくするプログラム!?(特集) ……2, 106
GRAPHMANを使ってみよう(特集)……2, 107
SXLIFEIIポップアップメニューの追加(特集) ……2, 116
コラム 「SXエンターテイメントキット」計画(特集) 2, 120
プロポーショナルピッチ文字表示 p_symbol()(ショート) ……2, 147
演奏バッチファイル play.bat(ショート) ……2, 148
コンピュータミュージック入門(特集)……3, 34
ミュージックツール実践活用(特集)……3, 37
Musicstudio PRO-68K ver.2.0(特集)……3, 40
MUSIC1.FNCで遊ぶ(特集)……3, 42
ヴァルナより町のテーマ(特集)……3, 47
大時計&ジャンしゅー(ショート)……3, 121
SXLIFEIIIライフゲームで姓名判断 ……3, 104
WINDOWシステム大比較 ……3, 100
迷路自動作成 rmaze3.X(ショート) ……4, 146
グラフィックデモ wak.c(ショート) ……4, 148
グラフィックデモ Forest.BAS(ショート) ……5, 141
まずはハードウェア環境の整備から(特集)……6, 44
三種の神器 DIR/CD/TYPE(特集) ……6, 46
COMMANDマスターへの道(特集) ……6, 50
SX-WINDOWで環境をつくること(特集) ……6, 57
基本はテキストファイル(特集)……6, 59
ダーツゲーム DARTS.BAS(ショート) ……6, 171
X-BASICの基礎(特集) ……7, 74
カットファイルを作成しよう(特集)……7, 77
MMLを画面に表示する(特集) ……7, 80
スプライトを加工する(特集)……7, 85
X-BASICでMAGICを(特集) ……7, 89
日付活用 ohayo.x(ショート) ……7, 143
出力デバイスを探る(特集)……8, 82
基礎からのカラー印刷(特集)……8, 88
HighFidelityへの挑戦(特集)……8, 95
PostScriptとはなにか(特集) ……8, 97
TeXからのアプローチ(特集) ……8, 99
IOCS用FONT200書体(特集)……8, 105
アクションゲーム UPDOWN.BAS(ショート) ……8, 131
グラフィックデモ MAND.BAS(ショート)……8, 132
MAGICの原理と活用(特集) ……9, 50
グラフィックパッケージMAGIC ver.2.0(特集) ……9, 52
3D関数の基本操作(特集) ……9, 72
MAGIC.FNCとMAGICダンプリストの入力法(特集) ……9, 77
縮小印字 SPRN.C(ショート) ……9, 148
マシン語の考え方(特集)……10, 74
IOCS,DOSコールの考え方(特集) ……10, 77
デバッガにて理解されたし(特集)……10, 82
避けて通れぬ道, アドレッシング(特集)……10, 90
実践アセンブラプログラミング(特集)……10, 94
ファイル名で遊ぶ EXACT.C(ショート)……10, 128
1行ブロック崩し LB_ATTACK.C(ショート) ……10, 129
ディレクトリ逆行 BD.X(ショート) ……11, 100
音とはなにか(特集)……12, 74
冬の夜長のスペクトル解析(特集)……12, 76
FM音源の波形を創る(特集) ……12, 81
Z-MUSIC公式ガイドブック(特集) ……12, 90
Z-MUSIC LIVE SHINDO ON STAGE(特集) ……12, 93
MIDIをめぐる環境(特集)……12, 99
MIDI出力方法論(特集)……12, 101
タイピング練習 KPP.BAS(ショート) ……12, 150
デモ なさけない★星★ PART2(ショート) ……12, 151
X68000マシン語プログラミング ……(→連載)
大人のためのX68000 ……(→連載)
ようこそここへC言語……(→連載)
吾輩はX68000である ……(→連載)
よいこのSX-WINDOW講座 ……(→連載)
Computer Music入門 ……(→連載)
Oh!X Live in '91 ……(→連載)
清水和人流プログラミング道場……(→連載)
DōGA・CGアニメーション講座 ……(→連載)
ハードウェア工作入門……(→連載)
シミュレーションプログラミング入門……(→連載)
CARDDRV用カードゲーム ……(→連載)
PASCALプログラミングへの招待 ……(→連載)
X-BASICプログラミング調理実習 ……(→連載)

イベント/ギャラリー

イベント
第3回アマチュアCGAコンテスト入選作品発表……4, 33
C-TRACE CGコンペティション ……6, 84
シャープパソコンフォーラム'91, マイクロコンピュータショウ'91&第72回ビジネスショウ ……7, 25
CG OSAKA'91, μイメージシンセサイザー ……8, 73
第3回サイクロンCG大会 ……11, 28
エレクトロニクスショウ&データショウ……12, 28

Oh!X Graphic Gallery
DōGA・CGアニメーション講座……1, 36

DōGA・CGアニメーション講座……………………………3, 25
DōGA・CGアニメーション講座……………………………6, 81
メタボール版C-TRACE TP＋ ……………………………6, 86
DōGA・CGアニメーション講座……………………………9, 25
DōGA・CGアニメーション講座……………………………11, 32
Oh!X Reader's ぎゃらりぃ
あけましておめでと一のコーナー……………………………3, 28
カラーイラスト大集合……………………………5, 36
THE USER'S WORKS
ガイアの牙……………………………2, 25
PARORAN & Lydion……………………………3, 26
マルチウィンドウシステム System-7C……………………………6, 88
詳説System-7C……………………………7, 47
Questland Stories/Ultimate Magic ……………………………9, 28

製品紹介

ハードウェア
光磁気ディスク CZ-6MO1 ……………………………1, 34
48ドット熱転写カラー漢字プリンタ CZ-8PC5 ………3, 30
X68000 SUPER内蔵増設ハードディスク CZ-68H ……3, 31
X68000キーボードチューンアップ……………………………3, 158
SOUND CANVAS SC-55……………………………4, 166
X68000 XVI/XVI-HD……………………………5, 40
SOUND CANVAS SC-55……………………………8, 149
X68000用3.5インチFDD TS-3XR1……………………………8, 150
V70AFPPアクセラレータボード……………………………12, 48
ソフトウェア
CARD PRO-68K ver.2.0……………………………3, 31
Musicstudio PRO-68K ver.2.0 ……………………………3, 31
C-TRACE68＋……………………………3, 94
Easypaint SX-68K ……………………………8, 76
NEW Print Shop PRO-68K ver.2.0 ……………………………9, 24
Musicstudio Mu-1 Super ……………………………9, 46
F-CARD GT……………………………11, 44

INFORMATION

ペンギン情報コーナー
X68000 SUPER(シャープ) ……………………………1, 176
書院 WD-A321/341/630/730(シャープ)……………………………1, 176
ファインスキャナ-X68(HAL研究所) ……………………………1, 176
RGB→S端子変換アダプタ XAV-1S(電波新聞社)……1, 177
バトルシート MENKURI(アイアンクラフト)……………1, 177
第5回ファンタジー＆ロールプレイングフェア ……1, 177
イマジニア協賛 杉本彩コンサート……………………………1, 177
OS-9/68000マルチユーザーシステムガイド ………1, 177
高速プリンタ搭載の書院 WD-A351(シャープ)………2, 158
フルカラーファクシミリ JX-5000(シャープ)…………2, 158
音声画像処理 THE 近江商人(ビットアート)……………2, 158
ハイパー電子手帳用ICカード「PA-9C30/5C01/5C02」(シャープ)……………………………2, 158
CCITT V.42bis搭載モデム MD96FS5V(オムロン) ……2, 159
AMIGA内蔵CDROM CDTV(コモドール) ……………2, 159
データバンクウォッチ VDB-1000(カシオ計算機) …3, 160
高速モデム MultiModemV32S(アメリカン・テクノロジー・グループ)……………………………3, 160
画面表示制御LSI V9990(アスキー)……………………………3, 160
中古パソコンの売買をパソコン通信で ……………………3, 160
電子手帳向け名刺データインプットサービス ………3, 161
LAN Expo'91……………………………3, 161
C-TRACE CGコンペティション……………………………3, 161
高速関数電卓 EL-546D/540D/506D(シャープ)…………4, 168
プリンタ切り替え器 Auto Boy 4ch(八戸ファームウェアシステム)……………………………4, 168
電子手帳用「松本亨の凱旋門」(水谷電機工業) ……4, 168
フロッピーディスク マークQ・クリエイトシリーズ(住友スリーエム)……………………………4, 169
フロッピーディスク スーパーPROシリーズ(化成バーベイタム)……………………………4, 169
Final SuperPack(エー・エス・ピー) ……………………………4, 169
CD-I内部OS CD-RTOS ver.1.1(マイクロウェア・システムズ)……………………………4, 169
アスキーネット法人会員制度(アスキー) …………………4, 169
電子手帳 PA-X1(シャープ) ……………………………5, 164
高精細液晶ビジョン XV-S1Z(シャープ) ……………5, 164
MIDIによる電源コントロール Maddie Rockeyシリーズ(ステージインスツルメンツ)……………………………5, 164
電子手帳用カード PA-3C25/32/33,7C1A/10A/21(シャープ)……………………………5, 164
ICメモリカードBIOSの開発(シャープ) ……………………5, 165
振動板素材「ホロファイン」(シャープ)……………………5, 165
NEW TAKERU(ブラザー工業) ……………………………5, 165
書院シリーズ WD-SD70/A810(シャープ)……………………6, 176
フィルムレコーダ FR-3000(日本アビオニクス) ……6, 176
コンパクトモデム MD24FB5V(オムロン) ……………6, 176
Auto BoyII(八戸ファームウェアシステム)……………6, 177
アイデアコンテスト(オムロン) ……………………………6, 177
JACA日本イラストレーション展(国際芸術文化振興会)……………………………6, 177
BOOK All in Noteの世界(KDDクリエイティブ) ………6, 177
イメージスキャナ JX-220X(シャープ) ……………………7, 160
カラーイメージジェット IO-735X-B(シャープ) ……7, 160
X68000用メモリボード KGB-X68PRKII(計測技研) …7, 160
ワイヤレスプリンタターミナル PRINTER MATE(積水化学工業)……………………………7, 160
電子手帳用通信カード＆モデム(シャープ) …………7, 161
文字放送活用情報システム TU-TX1＆PA-9C81(シャープ)……………………………7, 161
Floptical Disk(日立マクセル) ……………………………7, 161
写ルンです 防水(富士写真フイルム)……………………7, 161
X68000用3.5インチFDD X6835-2F(ファーベル)……8, 160
書院シリーズ WD-A520/540/550/560(シャープ)……8, 160
ハイパー電子システム手帳 PA-9550(シャープ) ……8, 160
電子手帳用プログラムBASICカード(シャープ) ……8, 160
電子手帳用ICカード PA-3C34/36(シャープ)……………8, 161
体験版ソフト付きフロッピーディスク SUPER PROシリーズ(化成バーベイタム) ……………………………8, 161
EYE-NET 音声/機械翻訳サービス(フジミック) ……8, 161
東映TVヒーローフィルムマラソン(東映ビデオ) ……8, 161
ワンタッチ用紙切り替え式プリンタ VP-870/1700(セイコーエプソン)……………………………9, 164
ビデオプリンタ VP-8000/8100(富士写真フイルム)…9, 164
高精細液晶プロジェクタ CU-SX1(シャープ) ………9, 164
OS-9用 Microware/X windows(マイクロウェア・システムズ)……………………………9, 165
学生情報通信論文ISID賞設立(電通国際情報サービス)……………………………9, 165
ヒューマン・クリエイティブ・スクール ……………9, 165
SAPPORO CG'91(北海道コンピュータグラフィックス協会)……………………………9, 165
BTRONパソコン 1B/NOTE(パーソナルメディア)……10, 160
3.5インチMO MOS300E(オリンパス光学工業) ……10, 160
薄型ケース付き3.5FD ニュー「スリム」シリーズ(富士写真フイルム)……………………………10, 160
X68000用ソフトウェア SPEAK SYSTEM/FUNCTION CALL(サザンエンタープライズ)……………………………10, 160
F1をデザインした写ルンです(富士写真フイルム) …10, 161
アスキーネットサービス拡大(アスキー) ……………10, 161
サウンドコンテスト(日本オーディオ協会) …………10, 161
第1回X68000芸術祭地区予選大会(シャープ)………10, 161
BTRONマシン MCUBE(パーソナルメディア)…………11, 160
10.4型TFTカラー液晶 LC-10C1(シャープ) …………11, 160
イメージスキャナ JX-320(シャープ) ……………………11, 160
X68000用拡張ボード SH-6BN1/U1/G1/F1(アイ・オー・データ機器)……………………………11, 160
電子手帳用ICカード PA-3C37/40/5C04(シャープ) …11, 161
NIFTY-Serve,JALNETゲートウェイサービス開始(ニフティ,アクセス国際ネットワーク)……………………………11, 161
X68000芸術祭東北地区大会(シャープ)…………………11, 161
書院シリーズ WD-A561/551/541/521(シャープ) ……12, 162
ノートブック型UNIX WS UN-10(シャープ) ……………12, 162
ポケットサイズモデム MD24FP5V(オムロン) ………12, 162
電子手帳用ICカード PA-9C2/9C31/32/38/5C05(シャープ)……………………………12, 162
MENKURIマイナーチェンジ(アイアンクラフト) ……12, 163
8.4インチTFTカラー液晶搭載のノートパソコンを開発(シャープ)……………………………12, 163
X68000芸術祭各地区大会(シャープ)……………………12, 163
FILES Oh!X 新刊書案内
エレクトロニクスキーワード集 ……………………………1, 178
東京都市学校2 バッドシティの快楽学……………………1, 179
柔らかい機械 思考のメカニズムの追求……………………1, 179
NTTが核攻撃される日 ……………………………2, 160
幻想としての文明 ……………………………2, 161
電脳と頭脳 ……………………………2, 161
地球56の顔 ……………………………3, 162
レイアウトひらめき事典 ……………………………3, 163
情報社会の弱点がわかる本 ……………………………3, 163
ワールド・エンド・ガーデン ……………………………4, 170
ポスト情報社会の到来 ……………………………4, 171
ニューロコンピュータ ……………………………4, 171
パソコンと私 ……………………………5, 166
図説 飛行術入門……………………………5, 167
頭を使いこなす マンダラ・メモ術 ……………………5, 167
ハイ・イメージ・ストラテジー ……………………………6, 178
メカノ ……………………………6, 179
科学なるほど事典 ……………………………6, 179
メディア・レイプ ……………………………7, 162
ハイパーアートの解剖学 ……………………………7, 163
大事なことはみーんな猫に教わった ……………………7, 163
人工現実感の世界 ……………………………8, 162
2000年のコンピュータ社会を読む ……………………8, 163
TwiLight Review 1 ……………………………8, 163
未来史の脳人都市 ……………………………9, 166
デジタル・ナルシス ……………………………9, 167
ザ・ベストゲーム ……………………………9, 167
情報新人類の挑戦 ……………………………10, 162
電脳騒乱節 2 ……………………………10, 163
フリッパーズ・テレビ ……………………………10, 163
ディファレンス・エンジン ……………………………11, 162
古代トーキョー大発掘 ……………………………11, 163
アポクリファ ……………………………11, 163
カッコウはコンピュータに卵を産む(上) ……………12, 164
独創的技術者の条件 ……………………………12, 165
笑うコンピュータ ……………………………12, 165

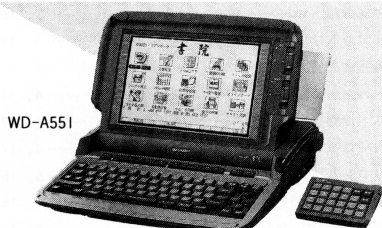
その他
Oh!X INDEX'90……………………………12, 158
常設コーナー
愛読者プレゼント
ペンギン情報コーナー
FILES Oh!X
Oh!X質問箱
STUDIO X
編集室から/DRIVE ON/ごめんなさいのコーナー/SHIFT BREAK/microOdyssey
特別付録
5″2HDディスク謹賀新年PRO-68K(1月号)
5″2HDディスク黄金週間PRO-68K(5月号)
X-BASICリファレンスブック(7月号)
X68000 THE GAME SOFTWARE BEST SELECTION(12月号)

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

はがき図形ソフト内蔵書院
WD-A561/551/541/521
シャープ


WD-A551

シャープは，はがき図形ソフト「はがき屋太郎」を内蔵し，「10キーステーション」で手元から編集，印刷ができる日本語ワープロを発売する。

新しく搭載されたはがき図形ソフト「はがき屋太郎」は，図形モードをまったく意識することなくワープロと対話する感覚で，簡単に素早く図形入りの年賀状や暑中見舞などを作成できる。画面上のキャラクター「はがき屋太郎」の，はがきの縦横，レイアウト，図形，賀詞，添え書きなどという質問に答えるだけで，年賀状や暑中見舞が出来上がる。しかも，WYSIWYG機能により，印刷されるイメージを画面で確認しながらレイアウト変更ができる。

そのほかの機能は従来の書院シリーズとほぼ同様になっている。

「WD-A561」は「4書体内蔵スーパーアウトラインフォント」(明朝，毛筆，ゴシック) 丸ゴシック」を搭載していて，毎秒117文字の高品位，高速印刷が可能。また，「WD-A541/521」は本体色にそれぞれ“インテリジェンスユーロブルー”，“エレガンスパールホワイト”を採用している。表示画面は「WD-A561/551/541」がハイコントラスト白黒液晶，「WD-A521」がブルーモード液晶を搭載している。

価格は「WD-A561」が250,000円，「WD-A551」が220,000円，「WD-A541」が200,000円，「WD-A521」が180,000円となっている（いずれも税別)。

＜問い合わせ先＞
シャープ㈱ ☎03(3260)1161，06(621)1221

ノートブック型UNIX WS
UN-10
シャープ


UN-10

シャープは，UNIXワークステーションとして最小，最軽量を実現した，ノートブック型ワークステーション「UN-10」を発売した。

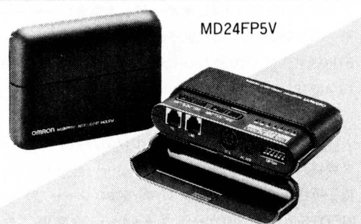
「UN-10」はA4ファイルサイズで本体重量が3.4kgと，ワークステーションでありながらノートパソコンにひけをとらない収納性と省スペース性を実現している。

OSにはUNIXシステムVリリース4.0および，それをベースにして同社独自の日本語/ビジネス環境強化を図ったOA/UX4.0を搭載。また，X-Windowバージョン11リリース4も搭載している。

CPUは20MHzのMC68030で，メインメモリ4Mバイト，40Mバイトのハードディスク内蔵モデル「CO-8001」が798,000円，メインメモリ8Mバイト，120Mバイトのハードディスク内蔵の「CO-8002」が1,198,000円となっている（ともに税別)。

＜問い合わせ先＞
シャープ㈱ ☎03(3260)1161，06(621)1221

ポケットサイズで低価格
MD24FP5V
オムロン


MD24FP5V

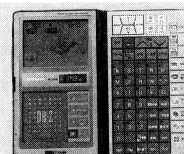
オムロンはポケットサイズで通信速度2400bpsのモデム「MD24FP5V」を発売した。

データ圧縮機能としてはMNPクラス5を搭載し，さらにCCITT V.42bisも搭載。エラー訂正機能としてもCCITT V.42bisとMNPクラス4を搭載している。電源にはACアダプタ，あるいは単3乾電池を4本使用し，アルカリ乾電池の場合約10時間，マンガン乾電池の場合約4時間の連続使用が可能。

本体には丸みを帯びたデザインを採用したり，コネクタ部のほこりの付着や静電気の悪影響を防止するフロントキャップ方式を採用するなど，携帯性と機能性も重視している。価格は36,800円（税別)。

＜問い合わせ先＞
オムロン ☎03(5488)3216

シャープ電子手帳用カード
PA-9C2/9C31/32/38/5C05
シャープ



電子手帳用カード

シャープ電子手帳用ICカードとして，以下の5機種が発売された。

○表計算カード128「PA-9C2」

ハイパー電子システム手帳「DB-Z」専用ICカードとして発売中の表計算カード「PA-9C1」の機能をそのままに，記憶容量が128Kバイトまで拡張された。記憶容量が2倍になったことで，大量のデータを一度に集計できるようになった。

価格は22,000円（税別)。

＜問い合わせ先＞
シャープ㈱ ☎03(3260)1161，06(621)1221

○仏和・和仏辞典カード「PA-9C31」

仏和辞典でフランス語約35,900語，和仏辞典で日本語約14,700語を収録。時事用語，料理用語のジャンル別の検索も可能。シャープの電子手帳すべてに装着が可能だが，

ハイパー電子システム手帳に装着した場合には，大画面を生かした見出し語リスト表示，画面タッチによる呼び出し/検索，逆翻訳機能なども使用可能になる。

価格は13,000円（税別）。

＜問い合わせ先＞

シャープ㈱　☎03(3260)1161,06(621)1221

○独和・和独辞典カード「PA-9C32」

独和辞典でドイツ語約34,700語，和独辞典で14,100語を収録。時事用語，日常用語のジャンル別の検索も可能。そのほかの機能は「PA-9C31」と同様。

価格は13,000円（税別）。

＜問い合わせ先＞

シャープ㈱　☎03(3260)1161,06(621)1221

○オセロ＆２ゲームカード「PA-3C38」

幅広いファン層を持つオセロを電子手帳用ICカード化。強さは４段階の設定が可能。ゲームは対コンピュータ，対人間のどちらでもできる。さらに，ペアマッチ（神経衰弱）と15パズルも楽しめる。

＜問い合わせ先＞

㈱バリエ　☎03(5272)3535

○京都財テク殺人事件カード「PA-5C05」

山村美紗原作の推理小説「京都マネーゲーム殺人事件」を，シャープ電子システム手帳「PA-9500」シリーズ用にICカード化。操作はタッチパネルによる対話形式で，約100枚の絵を見ながら，謎を解き明かしていく。ストーリーは第１章から第９章で構成されており，各章をクリアすると表示される４桁のパスワードにより（あるいはゲーム中に表示される36桁のパスワード）ゲームの続行が可能。

価格は7,700円（税別）。

＜問い合わせ先＞

㈱ヘクト　☎03(5275)5481

マイナーチェンジしたバトルシート
MENKURI
アイアンクラフト



MENKURI

昨年の暮れよりアイアンクラフトから発売されている，バトルシート「MENKURI」がマイナーチェンジした。

まず，“小さくて使いづらい”ということを解消するために，背もたれが追加された。体とジョイスティックの間に空間ができることで，ゆったりと操作することを可能にしようということである。

また，“低くて首が痛くなる”ことを解消するために，キャスターユニットを装着。座面の高さが60mm上がる。

この背もたれやキャスターユニットは旧タイプのユーザー向けに別売もされる。

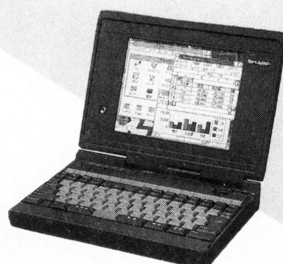
価格は本体（背もたれ，キャスターユニット付き）が13,600円，別売の背もたれ，キャスターユニットがそれぞれ4,800円，1,000円となっている（いずれも税込）。

＜問い合わせ先＞

アイアンクラフト　☎0256(33)6111

INFORMATION

8.4インチTFTカラー液晶搭載
ノートパソコンを開発
シャープ



シャープは，TFTカラー液晶ディスプレイを搭載したノートパソコンを開発した。

今回開発されたノートパソコンは，同社10.4インチTFTカラー液晶ディスプレイと比べて，1/3以下の低消費電力（６W）と，1/2以下の薄型化（12mm）を実現した8.4インチTFTカラー液晶ディスプレイを採用している。このノートパソコンはVGA表示のIBM PC/AT互換32ビットノートパソコンとして商品化が予定されている。

＜問い合わせ先＞

シャープ㈱　☎03(3260)1161,06(621)1221

X68000芸術祭各地区大会

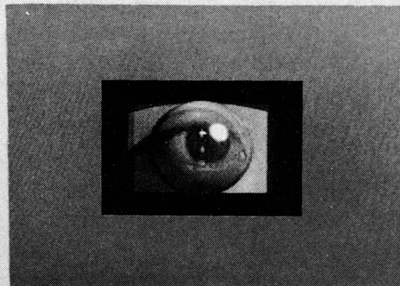
X68000芸術祭地区大会が各地で続々と催されている。各地区大会の受賞作品を簡単に紹介していこう。

中国地区大会では，前田浩氏制作の「TV in TV」が大賞を受賞した。これはDōGA CGAシステムとレイトレーシングソフトを使用して作られたアニメーションで，ぐにゃぐにゃと伸び縮みした球がTVに変形し，その中から目玉が飛び出してくるという作品。この大会では藤本幹雄氏制作の「白セン菌くん」が「TV in TV」と最後まで大賞を争い，その結果，特別入選という枠が設けられ，「TV in TV」とともに全国大会へと出場できることになった。この作品は水虫のもととなる白セン菌の繁殖の様子をシミュレートした環境ソフトである。

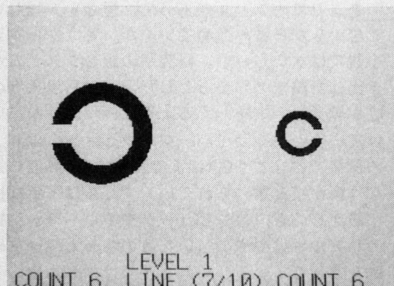
北関東大会の大賞に選ばれたのは，小林康弘氏制作の「Ｃ力検査」。視力検査をゲームとして遊んでしまおうというもので，画面上に「Ｃ」字型の輪が表示され，それに対応する方向のキーを押すという内容。両手でプレイするダブルモードや，タイムアタックモード，３人でプレイできる早押しモードなども用意されている。

神奈川地区大会の大賞を受賞したのは，文月涼氏制作のCGアニメーション「TORNADO」。第３回アマチュアCGAコンテストでも特別賞を受賞しているこの作品は，自分でデザインした車「TORNADO」のCMという形になっている。

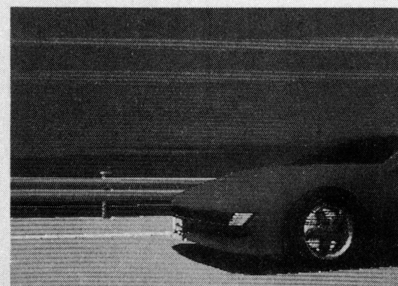
このように順調に進んでいるX68000芸術祭は来場者の数もいずれの大会でも多くなっており，まずまずの成功を収めているようだ。



TV in TV



C力検査



TORNADO

FILES Oh!X

このインデックスは，タイトル，注記——著者名，誌名，月号，ページで構成されています。早いもので，今年ももう残すところあとひと月。悔いのないよう，充実した日々を送ってくださいね。

参考文献
I/O 工学社
ASCII アスキー
コンプティーク 角川書店
テクノポリス 徳間書店
POPCOM 小学館
マイコン 電波新聞社
マイコンBASIC Magazine 電波新聞社
LOGIN アスキー

一般

▶CG CONSULTING
アニメ調のCGなどを描くときの，スキャナを使った原画の綺麗な取り込み法を解説。——リンダ霧山・バイオロボット柳，POPCOM，11月号，118-119pp.

▶アルゴリズムを見切ったぞ!?
グラフィックデータ圧縮アルゴリズム解説の3回目。——おにおん，テクノポリス，11月号，154-158pp.

▶日本パソコン百景
今回は群馬県太田市のPC-9801工場の見学に出かける。ここはPC-9801の実に半分を生産している。生産はほとんどロボットによって行われ，足りない部品の補給なども全自動だとか。——フデヨシ＆カワラ，ASCII，11月号，248-251pp.

▶パソコンで体験する天文学・宇宙の旅
SF「ロシュワールド」などに登場する双子の惑星。近接連星と呼ばれるこれらは卵型に歪んでいる。その理由を説明し，コンピュータでシミュレートを試みる。——福江純，ASCII，11月号，309-312pp.

▶The Play of Words
有名なことば遊びのひとつに「タブレット」というのがある。ある単語から単語までをひと文字ずつ取り替えながらつないでいく遊びだ。これをパソコンに解かせようというもの。——Hurtense Endoh，ASCII，11月号，313-316pp.

▶Micro MUSiQUES
コンピュータをライブで使う試みは年々進化し，乱数などの要素を曲の進行に組み込んだりするようになってきている。そういった新しい試みを取り入れたコンサートをレポートする。——編集部，ASCII，11月号，361-364pp.

▶TBN
最新CGフィルムが一堂に会するSIGGRAPH'91において発表された作品を写真で紹介。エンターテイメント，CM，理論解説のためのデモなど，多彩な顔ぶれが楽しめる。——編集部，ASCII，11月号，380-385pp.

▶欧州ハイテク事情
ヨーロッパ滞在の筆者が贈る各国のコンピュータ事情。今月はイタリアから，銀行のシステムについてお伝えする。コンピュータ化を進めている最中のイタリアでは知り合いの銀行員を作らないとやっていけない？——菊地薫，ASCII，11月号，400-401pp.

▶女性読者比率を10%にするためのページ
ワープロに関する新技術の巻。日立製作所のモードレス入力方式の仕組み，それから日本電気が発表した文書執筆支援システムについてレポートする。未来のワープロ像が見えるかもしれない。——編集部，ASCII，11月号，411-415pp.

▶MYCOM LET'S TAKE THE NEXT ONE
「選んで使える，フロッピーディスクの収納ケース」と題して防磁型・ケース型・ラック型などさまざまなフロッピーケースを一挙に紹介。——編集部，マイコン，11月号，127-130pp.

▶Mycom Soft Review
常駐型ステーショナリーソフト「Teleportion PRO-68K」を紹介する。電子手帳との接続・パソコン通信機能などの特徴を紹介し，使い勝手に批評を加える。——都築敏也，マイコン，11月号，153-157pp.

▶MYCOM WATCHING
四国香川にオープンしたレオマワールドを取材。東南アジアから中近東にかけての風景を模したテーマパークで，美術館や宿泊施設も用意されている。独自の作りをもった注目スポットだ。——菊地秀一，マイコン，11月号，190-193pp.

▶ビジネスマンのための情報管理術
シャープのハイパー電子手帳DB-Zシリーズに，新機種が加わった。従来機種の上位機種としてメモリ容量などの改良を受けたこのPA-9550を従来機種と比較しながら解説する。——塚田洋一，マイコン，11月号，232-235pp.

MZシリーズ

MZ-1500(BASIC MZ-5Z001)

▶ARMY BORST
あなたはテストパイロット。とはいっても，テストは実戦さながら。迫りくる壁と敵をよけ，時間内にミッションクリアせよ。タイム制のシューティングゲーム。——FROG，マイコンBASIC Magazine，11月号，122-123pp.

MZ-2500(BASIC-M25)

▶SUPER WARS
敵の惑星に乗り込み，敵の軍団をやっつける。敵機が13種類でオールBASICのシューティング。——もったんSOFT，マイコンBASIC Magazine，11月号，124-125pp.

▶LET'S PROGRAM
今月の宿題はちょっとしたパズル。隣り合う2つの数の和が平方数になるように数字を配置する。BASIC-M25で書かれた模範解答例が掲載。——藤本健，マイコン，11月号，271-279pp.

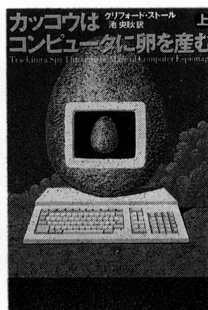
X1/turbo/Z

X1シリーズ

▶RALLY-Z
ドットイートタイプのカーレースゲーム。——大竹朗，マイコンBASIC Magazine，11月号，152-153pp.

▶CAT & MAUSE

新刊書案内



あまりにもスリルがあって，怪しい，怪しい，と思いながら読み進めていくが，どうやら本当に小説ではないらしい。研究所の課金システムが吐き出した請求のちょっとした矛盾から侵入者の存在を発見し，糸をたぐるようにハッカーを追いかけていくというサイバーなドキュメントは，上巻の最後でとうとうCIAにまで到達する。高度に知的な作業が積み重ねられていくスリルは下手な推理小説をしのぎ，コンピュータやネットワーク描写のリアルさはややこしいだけのネットワーク紹介本を凌駕している。それもそのはず，本書の著者は実際にハッカーを追跡した当人なのだ。リアルでないはずがないし，描写に破綻があるわけがない。ハッカーはGNUの欠陥を逆用して特権を獲得した，などというくだりはもうどきどきものである。さらに，ハッカーの手口が非常に高度なのだ。

ネットワークを調べ，侵入者がいつどこからハッキングしたか追跡していく面白さもさることながら，アメリカを縦横無尽に走るネットワークの実態をも見られる。その筋の人にはこたえられないお薦めの1冊（いや，上下で2冊か）。（K）

カッコウはコンピュータに卵を産む（上） クリストフォード・ストール著 池央耿訳 草思社刊 ☎03(3470)6565 四六判 302ページ 1,900円

書籍の価格は消費税込みです

2 OXFILE-12 0001 911031

大好物のチーズを探してチーズ屋に潜入。アクションゲーム。──金子秀樹，マイコンBASIC Magazine，11月号，154-155pp.

▶ネオ投稿プログラムコーナー
今月はダンプリスト打ち込みの間違いを探すときに便利な「メモリを拡大文字で」。雑誌のリストを指でなぞりながらちろちろ見れば，点検もはかどって効率アップ。──赤岩英明，マイコン，11月号，326・333pp.

X1turboシリーズ

▶満月の夜は喉が渇く
喉の渇きがいえるまで飲みつづけろ！ 水を飲む饅頭（？）が主人公の，なんともシュールなショートプログラムゲーム。──JIRONKA，マイコンBASIC Magazine，11月号，156-157pp.

X 1 turbo＋FM音源ボード（要NEW FM音源ドライバ）

▶Turbo New FM音源ドライバ作成プログラム
turboBASICで，NEW FM音源ドライバーを動くようにしようというもの。──山内千里，マイコンBASIC Magazine，11月号，185-188pp.

X68000

▶NEW PRODUCTS
パワーアップしたX68000用デザイン＆印刷ツール「NEW Print Shop PRO-68K Ver.2.0」の新しい機能などを紹介している。──6st.Inv，マイコンBASIC Magazine，11月号，76-79pp.

▶誌上公開質問状
付属ワープロでハガキ印刷はできるか？ ローランド社製CM-32Lを接続し使用するのに必要な機器は？ などの質問に解答。──多田太郎，マイコンBASIC Magazine，11月号，90-91pp.

▶BALLS
赤いボールをかわして，青いボールを取っていくゲーム。──久保繁雄，マイコンBASIC Magazine，11月号，158-159pp.

▶ENCLON
ミミズ型ロボット「エンクロン」で円を作ってボールを集めろ！──加藤淳一，マイコンBASIC Magazine，11月号，160-161pp.

▶ルート・ウォーカーズ
2人用の対戦スゴロクゲーム。──松本岳美，マイコンBASIC Magazine，11月号，162-164pp.

▶ミスティ・ブルー 〜I wanna close to you〜
エニックスのゲームミュージックプログラム。要NAGDRV＋MT-32系MIDI楽器。──中野和紀，マイコンBASIC Magazine，11月号，182-184pp.

▶X68000芸術祭インフォメーション
芸術祭会場に，満員神話生まれる！ 大入り満員の地区予選大会の模様をレポート＆受賞作品紹介。──山下章，マイコンBASIC Magazine，11月号，189-194pp.

▶今月の注目ソフト
360°大回転のアクションゲーム「キャメルトライ」が移植された。その出来具合はどうかな？──キャプション川口，マイコンBASIC Magazine，11月号，246-247pp.

▶SOFT EXPRESS
新作ゲームの紹介。ノア，ヴェルスナーグ戦乱，ディフレクター，エグゾンなど。──編集部，コンプティーク，11月号，86-87pp.

▶Hot Press
発売予定のパワーモンガーをいちはやくレポート。そのほか，プロサッカー68，全開電飾，NIKO²を紹介。──編集部，POPCOM，11月号，17-24pp.

▶こだわりレポート ドロボ〜の快感にヒタる！
つやつやのキャラクターが楽しいどろぼうアクションゲーム「ボナンザブラザース」を紹介。──ポンセ松崎，POPCOM，11月号，60-61pp.

▶ゲームの達人
深海ロボットアクション「アクアレス」をレビュー。──編集部，POPCOM，11月号，92-93pp.

▶ミュージック・パビリオン
思い出がいっぱい（らんま1/2のオープニングテーマ）ミュージックプログラム。──編集部，POPCOM，11月号，159-161pp.

▶GAMING WORLD
ワイヤー操作が特徴のアクションゲーム「アクアレス」，迷路を回転させてボールをゴールまで動かす「キャメルトライ」，キャラクターが可愛いアクション「フェアリーランドストーリー」など，新着ゲームを紹介。──編集部，テクノポリス，11月号，23-27pp.

▶NEW SOFT
発売予定のパワーモンガー，ディフレクター，シャングリラを紹介。──編集部，LOGIN，19号，18-31pp.

▶X68000新聞
年末に発売される予定の「出たな!! ツインビー」，「ジェノサイド2」などを紹介する。ほかに発売予定のパワーモンガー，飛翔鮫，コラムス，ディフレクターの紹介。第1回X68000芸術祭の模様レポート。──編集部，LOGIN，19号，230-235pp.

▶NEW SOFT
11月発売予定の飛翔鮫を紹介。──編集部，LOGIN，20号，24p.

▶最新ゲーム徹底解剖!!
アクアレスを攻略。──編集部，LOGIN，20号，140-143pp.

▶X68000新聞
木々を増やし森を育てるエコロジックなゲーム「ノア」や，シューティング「ラストバタリオン」，うさぎさんが可愛いキャラクターアクション「NIKO²」，などを紹介。──編集部，LOGIN，20号，242-245pp.

▶なんでもQ＆A
メインメモリ2MバイトでMultiwordを使っていたときに，スキャナ読み込みをしたら途中までしか読み込めない。解決法は？ などの質問に答える。──シャープ株式会社液晶映像システム事業部第2商品企画部，マイコン，11月号，368-369pp.

▶FREE SOFTWARE INDEX
最近どんなものが登場しているのか，編集部が主要ネットを調べた結果を一覧表示する。X68000にも多数のPDSが紹介されている。──編集部，ASCII，11月号，422-427pp.

▶長期ロードテスト
X68000EXPERTを使う担当者のレポート。SX-WINDOWを使っている担当者はスピードとメモリにそろそろ限界を感じている様子。Multiwordも試用体験しているが，スピードが遅いのはしようがないとして最終出力が美しい点に一応の評価を与えている。──編集部，ASCII，11月号，437-439pp.

▶GAME BOX
エグザクトの新作「アクアレス」，シャープの「ボナンザブラザーズ」を紹介。──SINON，伊藤ゆう，I/O，11月号，100-101pp.

▶ESE FIGHT
アセンブラを使ったゲーム。フィールドの中で相手のファイターを打ち落とすのだ。ちなみに対戦プレイのみ。──伊藤ゆう，I/O，11月号，156-157pp.

▶SOFT BOX
誰でも使えるX68000用印刷ツール「NEW Print Shop PRO-68K」がバージョンアップ。その内容を説明し検討する。──伊藤ゆう，I/O，11月号，166-167pp.

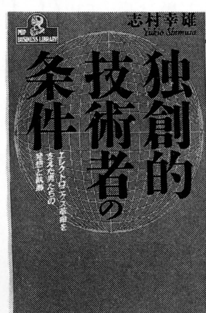
ポケコン

▶誌上公開質問状
通信販売でポケコンを購入したいのだが，どうすればいいのか？ カラー液晶表示のポケコンを出す予定は？ などの質問に答えている。──ドラゴン，マイコンBASIC Magazine，11月号，91p.

PC-E500

▶PIPERIS
落ちてくるパイプブロックを積み重ねて，ノルマの数だけ水を通す。アクションパズルゲーム。──駒野浩志，マイコンBASIC Magazine，11月号，166-167pp.

▶ASTRA VALLEY STORY
悪魔に狙われた谷を救え。ロールプレイングゲーム。──森内俊爾，マイコンBASIC Magazine，11月号，168-171pp.



独創的技術者の条件

最先端と呼ばれる技術は，秒進分歩といわれるほどに加速度的な発展を遂げている。なかでも，戦後の日本のエレクトロニクス技術の進歩には目をみはるものがある。本書はこのエレクトロニクス分野の技術革新に着眼し，IC，パソコンなど15項目を取り上げ，それらの開発の背景や経緯をまとめたものだ。欧米で生まれた原理や基本機構を日本がどのように応用研究し，「技術大国」と呼ばれるに至ったかを知ることができるだろう。

志村幸雄著　PHP研究所刊　☎03(3239)6221　新書判　254ページ　850円



笑うコンピューター

コンピュータ人間へのインタビュー，逸話，物語などさまざまなコンピュータ話をまとめた本。数百人ものコンピュータ人間に取材したり，映画の中のコンピュータを考えたり，はたまた自分が感じたコンピュータ像からお話を作ってみたりと，いろいろな方面からコンピュータをとらえている。アメリカ人っぽい彼女のウィットに富む文章も，軽く読ませるのを手伝っていてよい。

カーラ・ジェニングス著　邦つゆこ訳　技術評論社刊　☎03(32225)3293　四六判　271ページ　1,500円

書籍の価格は消費税込みです

QUESTION and ANSWER



この手の質問には，もううんざりされているかと思いますが，死活問題なので聞かせていただきます。使用機種はX68000です。

このあいだ，見事にデータの入ったディスクをフォーマットしてしまいました。でも，Cスイッチを付けてましたので消えたのはFATだけだと思うのです。だからなんとか復活が可能ですよね？ こういうときはデータ内容がディスクのあちこちに散らばっていて，繋がりを見つけるのが超大変だと本で読んだことがあります。が，今回のデータはほとんどが行番号付きのBASICプログラムですので，それはけっこうラクだと思います。そこでディスクの各セクタの内容を画面に表示（ダンプ）して，ほかのディスクへセーブするようなプログラムを作成していただきたいのです。BASICじゃディスクの任意のセクタを読むなんてことはできないので（私はBASICしかわかりません）。「お前な，バックアップくらいとっとけよ」といわれそうですが，そのバックアップをするときにオリジナルのほうをフォーマットしてしまったんですよぉ。

大阪府　中西 道一



今後このような間違いを起こさないためにも，マスターディスクにプロテクトシールを貼ってからバックアップをとるようにしたいですね。マスターをフォーマットしたとはいえ，この場合はCスイッチを付けたようですから，ディレクトリとFATの内容が失われただけで，データはディスクの中にまだ残っています。

質問を読むと，中西さんはディスクの管理がセクタ単位で行われていることは知っているようですね。Human68kやMS-DOSでは1セクタが1024バイトと決められています。ですから，仮にファイルサイズが2200バイトのファイルがあったとすると，このファイルは3セクタにまたがって保存されることになります。

このときにセクタ0→セクタ1→セクタ2と順番に使われるのなら，先頭のセクタ番号さえわかればいいのですが，実際にはファイルの削除や作成を頻繁に繰り返したディスクでは，ディスクの空きスペースを有効に使うために，データがディスクのあちこちに分散するようになります。

そのためにセクタのつながりを記憶しておく場所を設けて，ファイルの読み込みはそれを頼りに行うようになっています。その記憶場所を私たちはFAT（File Allocation Table）と呼んでいます。では，ファイルの大きさが1024バイト以下のときはどうでしょう。これなら1セクタにデータが収まりますから，ディスクの復旧作業もさほど苦にならないと思います。

さて2HDディスクの場合，データエリアはセクタ12（1セクタ目をセクタ0として）から始まります。このセクタ12をデバッガを使って読み込む方法を紹介しましょう。まずデバッガを起動してください。カーソルが点滅して入力待ちになっていますね。デバッガにはディスクの内容を直接読み込むコマンドとして，

R@＜アドレス＞＜ドライブ番号＞＜レコード番号＞＜レコード数＞

が準備されています。アドレスにはディスクから読み込むデータを格納する先頭アドレスを指定します。このときOSやデバッガが使用しているアドレスを指定してはいけません。デバッガからPコマンドを実行して，

user program from $????????

で表示されるアドレス以降を指定します。ドライブ番号はドライブAなら1，ドライブBなら2，以下Cが3，Dが4……という具合に指定します。

さらにセクタ番号，レコード数を16進数で指定しますが，セクタ番号は1セクタ目が0となっていることに注意してください。するとドライブAに入っているディスクのセクタ12を読み込むには，

R@ c00000 1 c 1

のようにすれば，C00000$_H$からセクタ12の内容が読み込まれることになります。ちなみに格納アドレスであるC00000$_H$はグラフィックRAMの先頭アドレスです。グラフィックRAMをRAMディスクとして使っている人は，RAMディスクの内容が破壊されてしまうので別のアドレスを指定するか，RAMディスクドライバを外したシステムで起動し直してください。

グラフィックRAMを格納アドレスにすれば，読み込めるレコード数は最大512（1セクタ＝1Kバイトだから）になります。

また，デバッガにはメモリ内容をディスクに書き込むコマンドも用意されています。これは，

W＜ファイル名＞，＜先頭アドレス＞，＜最終アドレス＞

です。前に読み込んだセクタ12を別のディスクにセーブするなら，

W b:sec12,c00000,c003ff

のようにします。

リスト1

```
 10 /* save"e:¥qa12.bas"
 20 /*
 30 int ai,bi,ci
 40 char rec(1023),err=0
 50 /*
 60 ai=fopen("g:back1","r") /* 読み込みファイル
 70 bi=fopen("g:text","c")  /* 書き込みファイル
 80 /*
 90 repeat
100 ci=fread(rec,1024,ai)
110 err=0
120 for i=1 to 127
130  if rec(i)=&HE5 then {
140  if rec(i-1)=&HE5 then {
150  if rec(i+1)=&HE5 and rec(i+2)=&HE5 then err=1:break }}
160  if rec(i)<&H20 then {
170  if rec(i)=9 or rec(i)=10 or rec(i)=13 then continue else{
180  if check_kanji(rec(i-1))=0 then continue else {
190  err=1:break }}}
200 next
210  if ci<>0 and err=0 then {
220  for i=1 to 1023
230   if rec(i)=&H1A then {
240   if check_kanji(rec(i-1))=0 then continue else {
250   rec(i)=&H20:break }}
260  next
270  fwrite(rec,ci,bi)
280  fputc(&HD,bi):fputc(&HA,bi) }
290 until ci<>1024
300 fcloseall()
310 end
320 /*
330 func check_kanji(a;char)
340   if (a>=&H80 and a<=&H9F) or (a>=&HE0 and a<=&HFF) then {
350   return(0) } else {
360   return(-1) }
370 endfunc
```

ここまでの説明でデバッガさえあれば，手間はかかろうともフォーマットしたディスクからデータを取り出し，ほかのディスクにセーブできることがわかったと思います。

中西さんの場合は行番号付きのBASICプログラムが多いようですから，データを取り出すことさえできれば復活させることもそう難しくないでしょう。それ以外の実行形式のファイルなどの復活は相当の根気と勘を必要としますから，自信がなければ素直に諦めてください。

さて，ドライブAに壊れたマスターディスク，ドライブBにフォーマットされたバックアップディスク，グラフィックRAMをラムディスクなどで使用していないのなら，

```
R@ c00000 1 c 200
W b:backup1,c00000,c7ffff
R@ c00000 1 c+200 200
W b:backup2,c00000,c7ffff
R@ c00000 1 c+400 c4
W b:backup3,c00000,c30fff
```

で，マスターディスクのすべてのデータをドライブBに取り出すことができます。

次にこのファイルからテキストのみを取り出します。そのためのプログラムをリスト1に紹介します。にわか作りですが，結構きちんとテキストのみを取り出してくれます。読み込み，書き出しファイル名は60, 70行を変更してください。もちろん，コンパイルしたほうがいいことはいうまでもありません。

さて，テキストのみのファイルができたら，それをエディタに読み込みます。この時点ではセクタが先頭から順番に使われていた場合を除いて，テキストがあちこちに散らばっているはずです。それをカット&ペーストを使ってテキストの繋がりを正しくします。最後にこのファイルをセーブして，めでたくファイルが復活したことになります。

なお，改行コードのないテキストを編集する場合は，起動時にMスイッチで1行の長さを128バイトにしておくと便利だと思います。また編集中にある部分だけを切り出したいときは，ESC+Wを使うと便利です。詳しい説明は，Human68kのユーザーズマニュアルに書かれていますから，そちらを参照してください。（影山裕昭）



コンピュータとはまったく関係ないのですが，なぜプリンタの3原色はCMY（シアン，マゼンタ，イエロー）なのでしょうか。色の3原色は赤，青，黄なので赤や青は再現が不可能だと思うのですが。また，光の3原色はRGB（赤緑青）なのに色の3原色では，その補色であるCMYではなく赤，青，黄なのでしょうか。おそらく，原子の振る舞いが関係していると思うのですが。
私にはわかりません。もし答えてくださるなら科学的にお願いします。変な質問でスミマセン。　石川県　佐渡　詩郎



「科学的に」というのは難しいかもしれませんね。まず，一般にいわれている3原色というものは，その色の組み合わせでほかの色すべてを表せるものではない，ということを知っておいてください。これらはあくまで疑似的なものです。

色の本質は光の周波数ですから，そもそも，色を混ぜてほかの色を作るというのは，原子の振る舞いというより，視神経の構造と視覚認知の問題にかかっています。

目には3種類の神経細胞があるといわれています。それぞれが，どのような役割かはさだかではありませんが，色を表す際にどうしても3種類の信号が必要であることから，3つの刺激の複合であると推測されています。

違う色の光を混ぜると色が変わるというのは光の周波数が変わるのではなくて，視神経の受け取る刺激バランスが変わるということです。

光の3原色RGBというのは，RGBそれぞれのもたらす刺激の組み合わせでかなり広い範囲の色をカバーできるということを意味しています。基本的にRGB以外の周波数の光でも適当に散らばった3種類を組み合わせればかなりの色を表せるはずなのです。

2種類でも可能です。カラーテレビの実用化の際にはまず2原色でいくか，3原色でいくかを検討したといいます（当然，表せる色数は違う）。

さて，このように広い範囲を表せるRGBをインクにして，その周波数を吸収させるとCMYになります。インクの3原色として知られています。プリンタもこれを使っていますね。CGや印刷でもすべてCMYを色の3原色として扱います。しかし，小学校以来，色の3原色は赤，青，黄だとも教えられています。なぜでしょう。

おそらくは図工ないしは美術の時間に使う画材（不透明水彩絵の具）のせいだと思うのですが……。絵の具の赤，青，黄を混ぜると普通，灰色になります。理論上は黒にならなければいけません。不透明水彩では色を混ぜても明度を下げることができないため，インクの3原色はそのままでは使えません。また，赤，青，黄だけではすべての色を作ることはできません。絵の具は赤，青，黄，黒，白の5原色を基本としていると考えてよいでしょう。

絵の具は，その色を特に強く反射し，インクはその色以外を吸収する，という違いにより混ぜたときの発色メカニズムがまったく変わってきているのです。

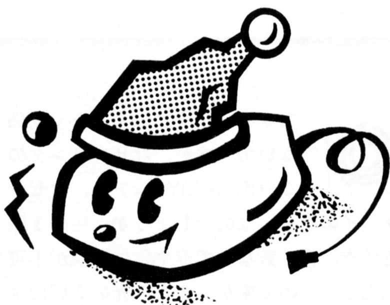
最初にいったとおり，色というのはかなり人間側で決めている概念です。同じエネルギーでもどの周波数帯であるかによって感じ方はまるで違います。「RGBを混ぜると白になる」というのも，もし太陽がK型恒星だったらまったく違った認識になったことでしょう。もし，人間の目が紫外線領域まで見えれば，現在のフルカラー画像は無茶苦茶な色バランスなのかもしれません。いずれにせよ，そういった人間の目のいい加減さが幸いして，いろいろな周波数帯の光をたった3種類の周波数で表すことができているわけです。（中野修一）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒108 東京都港区高輪2-19-13
NS高輪ビル
ソフトバンク株式会社出版部
「Oh! X質問箱」係

STUDIO X ON AIR



FROM READERS TO THE EDITOR

わあい，ジングルベール♪の季節がやってきました。寒いけどイベント好きの人は待ちに待ってた年末だ。そして，世間一般の人はちゃんと休暇を楽しんでいることでしょう。なんて羨ましがってなんかいられない。今夜も朝まで仕事でい。

◆特集のメインタイトルが「マシン語との邂逅」となっていた。「邂逅」とはなんや？ と思って「漢字林」を使って調べてみた。それによると「巡り会う，うちとけるさま」，となっていた。私もこれを契機にマシン語に出会い，うちとけたいと思った。　高坂　剛士(19)大阪府

僕は邂逅の文字を変換してくれないASKを横目にワープロ辞典を引いています。

◆現在X68000でアセンブラを勉強中の私にとって，10月号の特集はたいへんおもしろく，かつ，ためになる内容でした。私にとってアセンブラほど，魅力的な言語はありませんのでこれからも今回のような特集があると非常にうれしいです。　高島　和典(29)群馬県

がんばりましょう。

◆10月号の特集はよかったです。特にアドレッシングモードの説明が参考になりました。しかし，もっとアドレッシングモードの具体的な使い方があればさらによかったでしょう。説明だけならほかの本(68000プログラマーズハンドブックなど）で説明されているからです。こういうときにはこのアドレッシングモードを，こういう場合にはこのアドレッシングモードを使えばメモリ効率がよくてしかも速い。というようなプログラミング時に役立つような説明もほしかった。　塩田　耕司(22)広島県

塩田さんも手探りでいいからいろいろ試してみては？

◆学校の授業でアセンブラをやることになったのだが，テキストごとに想定するアセンブラの仕様が異なるため，最近混乱しそうです（インテルっぽいヤツ)。そんななかで，10月号の68000のアセンブラ特集。さらに混乱するかと思ったけど制約が少ないし，系統だった命令群は覚えやすく「ああ，美しい」などと感じてしまった。実際に活用できるようになるのは先の話だとしても，アセンブラ独特のおもしろさというものが，ほんの少しわかったような気がします。　佐藤　充浩(20)長崎県

あとは実践のみ。

◆そろそろ，アセンブラに手を出そうかと思っていたのでとても参考になる特集でした。まだ，わからないことがたくさんあるけど，一筋の光が見えてきたように思えます。中森章氏のC言語講座のおかげでC言語もできるようになってきたし（まだポインタについてわからないこともあるけど)，ABCを征服する日も近いかな。　武藤　一文(19)埼玉県

バリバリですね。

◆パソコンの周辺機器が充実し，プログラミング技術だけでなく，実際に周辺機器を操るための素養を必要とされている現在，新連載の「Creative Computer Music入門」はとてもタイムリーな記事でしょう。しかし，「詳しくは音楽の教科書を参考にしてください」というのはちょっと不親切だと思います。面倒でしょうけれどきちんと説明してほしかった。それから，マイナーのスリーコードが間違っているのも，ちょっと気になりました。ともあれ，瀧さんの連載には期待していますからね。　西村　進(21)京都府

きっと期待に応えてくれるでしょう。

◆最近C言語でプログラミングをしていても（公私ともに）#asm～#endasmがやたらと多い。「だったらアセンブラで組めば」と友人などにいわれるがCが好きなんだからしょうがない。それに，CC.Xはまだまだオプティマイズが甘いんですよね。でも好き。　寺田　泰(22)北海道

強いなあ。

◆やっとハードディスクを買いました。使い始めると，もうフロッピーディスクには戻れません。スピーディなディスクアクセスに大容量とくれば，もうたまらないものがあります。そうして私のパソコンライフはとても快適なものになりました。が，そのおかげで現在１日１食インスタントラーメンの毎日を過ごすはめに……。とほほ。　石田　智義(21)京都府

体に気をつけてね。

◆最近は市販のゲームはあまり購入しないで，もっぱらパソケットなどで同人ソフトを手に入れたりして楽しんでいます。ハズレをつかまされるときもありますが，出来のいいものも結構あります。特にちょっとした息抜き用のカードゲームは重宝しています。手軽なゲームが安く提供されるのは，とてもありがたいことです。　白井　宏尚(21)千葉県

自由な発想から生まれた，キラリと光る何かがありますからね。

◆「Might&Magic」を解き終えました。X68000を買ったときに一緒に付いてきたものなので，クリアまで１年半かかったことになります。パーティが全滅したあとはしばらくやらなかったし，魔法陣を解くためにも時間がかかったから，実質２カ月くらいかな。クエストもアイテムもモンスターも山ほどいて，ゲームの最中はこの世界に浸っていました。夜な夜なモンスターの悲鳴を聞きながらマッピングする姿はさぞかし異様だったんだろうな（誰も声を掛けてこなかった)。少し反省。　岩瀬　貴代美(19)福岡県

しばらくしたら「Might&MagicII」にはまってたりして。

◆Oh!Xはうちのお父さんが読んでいます。これから寒くなりますが皆さんがんばってください。あ～，もうすぐ私の足の指がしもやけでポンポンに腫れてしまう。　水谷　さよ(14)三重県

くう，人の情けが身に染みるぜい。

◆Oh!XとX68000は主人の宝物です。先日，X68000の調子が悪く１週間ほど修理に出しま



◀真鍋　博之　東京都
確かに，生のハガキをみるかぎりきれいに印刷できるものですね。でも，印刷されたときにこの細かいドットが再現できるかちょっと心配。



◀小林　貴洋　東京都
う～ん，まるで○原一至が描いたみたい。それとも岡○武士かな？ ネコとモドキに遊ばれる主人公の顔が，情けなくてとってもいいぞ。

した。平日には主人も会社があり，それほど気にならなかったようですがX68000不在の休日はかなり寂しい思いをしたようです。この休日はOh!Xを読みながら愛しのX68000の無事をひたすら祈る主人の姿がありました。

森本　幸子(25)千葉県

そんなご主人の姿を見てちょっぴりやけてしまいました？

◆主人のX68000でゲームをしていると「ちょっと貸してみろ」と言って取り上げられる。無視してひとりで遊ぼうとすると「誰の機械だと思っているんだ！」と怒られる。離婚するときはX68000をもらっていこうと考える今日この頃。

栗林　のぞみ(21)北海道

主婦は強し！

◆ターミネーター２のCGはすごかった。編集部の皆さんは観ましたか？ と〜ってもおもしろかったので，もしもまだ観てなかったらぜひ観にいってくださいね。加藤　真理子(18)愛知県

徹夜明けで映画を観にいったので，上映中ぐっすり寝てしまった。しょうがない，もう１回観にいこうかな。

◆最近はなぜかゲームをしなくなりました。では，パソコンで何をやっているかというと，もっぱら自作の競馬シミュレータを使って実践を兼ねた投資をしています。現在では改良につぐ改良でよく当たります。この間なんかは10万も当てることができたのですから，うまくいけばこの調子で今年の冬に出るかもしれない，X68000の新機種の資金にしてしまおうかと思っています。　鬼谷　秀樹(20)福岡県

そうは問屋が卸してくれる？

◆先日人通りの多い道を歩いていると，前方からムチャクチャ顔もスタイルもいい，ボディコンねーちゃんが歩いてきました。「おおっ」と思いしばらくじっと見つめていました。ここで理性を働かせてまわりを見回すと，男は必ずといっていいほどじろじろ見ているのです。男はやっぱりすけべだと改めて思いました。

倉知　和弘(16)北海道

そこはそれ「すけべは男の甲斐性だ」，ということだからね。でもずるい，それほどの美人なら僕も見たかった。

◆この間，ゴルフの打ちっぱなしに行ってきて以来，ゴルフにはまってしまった。さっそくクラブ２本（安物だけど）を買ってきてほとんど毎日素振りしています。そして，ゴルフって見た目よりハードなスポーツだなあ，と体感してからプロゴルファーたちの苦労が少しわかったような気がします。　宮沢　毅(21)長野県

金もヒマもない僕は「遥かなるオーガスタ」で気分だけゴルフの楽しさを味わっています。空しい。

◆注意！ 徹夜のパソコンは危ないです。先日徹夜で「イース」をクリアして頭がボーッとしていたため，顔でも洗ってシャキッとするか，と思い洗面所に立ち，洗顔フォームを手にとって顔を洗い始めたのです。すると，なんか肌触りが変だなと思ったので，ふとチューブを見るとなんと“アクア○レッシュ”だった。歯磨き粉を顔に塗るとべたべたして気持ち悪いよ〜。だから徹夜のパソコンは怖い。

山崎　勘太郎(19)愛知県

過ちは繰り返すといいます。今度は洗顔フォームで歯を磨かないように気をつけよう。

◆先日，思い切って光磁気ドライブを購入しました。なかなか動作は快適で満足していますが，やっぱり高いですね。最近では，HSTモデム，プリンタ，光磁気ドライブ，増設RAMボードなど，本体より周辺機器のほうがずっとお金がかかっているなあ。　堀　義弘(22)東京都

次はどんな周辺機器を狙っているのかな。

◆２カ月前にたたき売りされていたX68000PRO IIを買った新米者です。これまでパソコンを買ったことも使ったこともない僕は，学校でNEWSを使っています。うーむ，パソコンもろくに使えない人間がX-WINDOWでプログラムなんかしてもいいのでしょうか。

高橋　正樹(22)岩手県

逆に余計な知識に縛られることなく，すんなりコンピュータの世界に馴染めるかもしれませんよ。

◆台風は怖い。何が怖いって，瓦は飛ぶし家は揺れる。いま窓の外を見ると雨が真横に降っている。さっきは４分ぐらい停電だったし。こんな日はハードディスクが心配でX68000に触れないって感じです。　佐藤　正明(18)福岡県

毎年この季節になると出る話題。いままで台風で怖い思いをしたことがないから実感がわかないな。

◆転勤で茨城県は土浦近くの村に引っ越しました（これで市民から町民，ついでに村民になってしまった）。で，ヤケを起こしたわけではありませんが筑波山に行ってきました。そこにはやっぱりいましたねえ。さすがに大道芸人さんはいませんでしたが，ガマちゃんに油，日本刀に口上を書いた巻物まで。ほかには山頂近くまで行くロープウェイにケーブルカーがありましたが，あとは回転展望レストランぐらいで何もない感じです。しかし，回転展望レストランは自分が動いているか風景が動いているかわからなくなるという，不思議な感覚に陥ってなかなかおもしろかったです。　大島　靖浩(29)茨城県

不思議な感覚から気持ち悪くなり，せっかくの食事を戻してしまったらもったいないですね。

◆我が家では今年初めて夏のジャンボ宝くじを買いました。結果はハズレでしたが，１枚だけ100万円の当選番号が同じで，組違いというのがありましたので少し安心しています。なぜならいままで何をやっても運がなかったものが「ほんの少しかすめているなあ」ということがあって，「少しは運が上向きかな」と感じられたからです。しかし，１億もいらないからせめて元手だけでも返ってきてほしかった，というのが本音です。　藤原　彰人(21)岡山県

ま，そのうちいいことがありますって。

◆大学で民法の教授がディアマンテを買ったそうな。そして，ある日東京に行ってホテルに宿泊したとさ。朝起きて駐車場まで行ってみると，ディアマンテの４本のタイヤすべてが煙のように消え失せ，ディアマンテはごていねいにブロックの上に乗せられていたそうな。合掌。

野田　博(21)群馬県

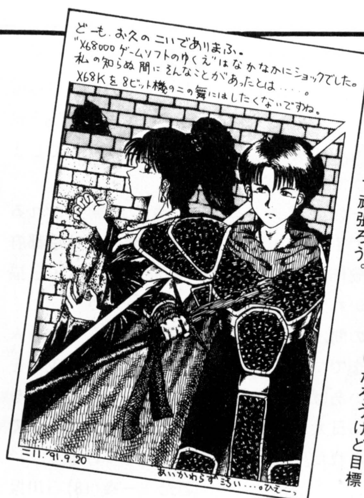
なんておちゃめな泥棒さん。それとも新手の嫌がらせかな。

◆三重県四日市市にマイコンテック四日市がオープンします。美人のお姉さんが店員として働いているので，ぜひご来店ください。目印は河内という名札です（おいおい）。

河内　久美(23)三重県

しかし，三重県はちょっと遠いですね。なんとか出張扱いで取材にいきたいけど。編集長，取材許可を！

◆もうすぐ冬がやってきます。それで困るのがコンセントの問題です。現在私の部屋にはX68000，CM-32L，HGS-X68，CZ-8PK7，PC2，コンパチプレイヤー，S-VHS，VHS，Hi-Fi，ROXY，X1turboZがあります。ということはコタツとファンヒーターのどちらかひとつを使うとしても，コンセントがひとつ足りない。そして，コタツに入りながらパソコンをフルに使うのは不可能だし。しかたないから今年の冬はファンヒーターの代わりにX68000の冷却ファンで我慢しなきゃ。　黒谷　雄二(20)栃木県



◀小井田　伸雄　東京都
お久しぶり。岩手の元気少年が東京の浪人となって帰ってきた。なんてね。つらいだろうけど目標に向かって頑張ろう。



◀小川　裕美　山口県
本当に出るのか？ いや，きっと出るだろう。の「出たな!!ツインビー」発売決定。これは期待したいものですね。

ビデオデッキの電源から暖を取るのもひとつの方法でしょう。

◆昔「プーさんに似てる」って言われたことがありました。別に太ってもいないし，似ても似つかないと思ったので気にもしませんでした。で，最近では貴花田に似てるっていわれることがあります。どうやら僕はプーさんに似てるみたい。　　西村　佳哲(21)京都府

ということは貴花田はプーさんに似ているわけですね。

◆10月1日にめでたく内定をいただきました。就職先は，産業ロボットを作っているメカトロ関係の会社です。振り返ってみれば小さい頃からマジンガーZなどのロボットヒーローものを見て育ち，中学の頃読んだ漫画の影響で高校では理系を選択して工業大学に進学し，結局ロボット屋さんに就職してしまった。そして，これからの人生（というとおおげさだけど）もロボットと一緒でしょう。皆さんも夢は持っていたほうが将来楽しいですよ。

本田　英雄(22)埼玉県

望んだ職業につけて幸せそうですね。

◆夏の終わりにニュージーランドへ行ってきました。澄み切った青い空，底の見える海，公園に行けば羊の群れがたくさんいるいる。鍾乳洞のツチボタルは星空のようで，まるで別世界にいるような感覚になりました。しかし，夕方5時には閉店する店が多く，物質文明の極致に慣れた人間にはちょっとつらいものがありました。中心部には「石を投げれば日本人に当たる」くらい日本人だらけ。視野を広げるいい機会になりました。　　内藤　陽一(24)愛媛県

いままでとはまったく違う環境を体験して何を学びましたか？

◆かつてライトサーベル（広告はサギだった）を所持していた私としては，スターウォーズに期待しています。そして，絶対にデススターの突入シーンでベン・ケノービがいうあのセリフがほしい。「ルーク・ユーズ・ザ・坊主」，坊主を使ってどうする。　　弦元　達也(20)香川県

そんな，東洋の神秘「お経パワー」を侮ってはいけませんよ。

◆どうも最近疲れているせいか，ちまたで話題の網タイツをはいた画王を見てしまった。そんなもん，いるわけないと思ってたのに。やっぱ，疲れているんだろうな。篠崎　篤史(24)静岡県

最近は疲れが溜まると突如踊りだしてしまう僕であった。

◆OH-Xというヘリコプターが今度開発されるそうです。はい。　　伴　哲也(20)京都府

機体の横にデカデカとSOFTBANKと描かれていたら笑ってやってください。

◆この間，秋葉原の路上で客引きをしている人に店までついていったら，店員にセリフ攻めをくらいあげくのはてに，浪人の身である自分を勝手に日大2年生と決めつけ息子の自慢話まで聞かされた。皆，客引きには気をつけよう。

酒元　一幸(18)石川県

危ないなあ。

◆山形県にもやっと新幹線がくる。それはいいのだが，そのために2カ月間電車が通らなくなる。現在，自転車で学校まで通うハメになった。バスもあるけど本数が少ないうえ，通学中酔いそうでいやなのだ。でも自転車はやっぱり疲れる。1時間以上かかるうえに道路はトラックがビュンビュン通る。今日もトラックが多くて風にあおられてこけそうになった。はっきりいって危ない。　　安藤　哲(17)山形県

毎日がスリルに満ちていて刺激にはことかかなさそう。

◆パソコンの楽しみのひとつにI/Oスロットを利用したさまざまな“オリジナル機能”の付加があります。特にオリジナルと呼んだのは自作するほうが買ってくるより作る楽しみがあるからです。ところがX68000のハード関係の資料を1年くらいあちこちで探しましたが，ほとんどありません。I/Oスロットのタオムチャートひとつ手に入らないなんて。これじゃ企業も研究室も買ってくれないと思う。

五十嵐　豊(24)千葉県

だめを承知でシャープに直接掛け合ってみるのもひとつの手ですよ。

◆C MAGAZINE 9月号の156ページによると，どうやらG＋＋の付録収録が間近らしい。GCC，G＋＋の次は当然TeXでしょう。さあ，皆でC MAGAZINEにハガキを書こう。

藤田　明(26)群馬県

宣伝，宣伝。

◆TETSUはやはりグレープフルーツ味ですね。鉄分とグレープフルーツの酸味がよくマッチしていて非常に……美味しくない。近所の薬局で売っていることを知ったときには驚きました。

溝口　信太郎(21)愛知県

薬局で売っているとは侮れませんね。やっぱり，あの鉄分が体にいいのかな。

◆聞いてください！ クリアできたんです。あの「パロディウスだ！」が。シューティングゲームが，苦手なくせに好きな私が初めてクリア。うれしかった。もう言葉に表せないくらいの感動です。ただしコンティニューしまくりでようやくですけど。このゲームは女性でも楽しめるゲームでしたね。　　馬場　基子(21)兵庫県

それはおめでとう。で，自慢ばかりされるのもしゃくなので僕も自慢しよう。聞いてください。ようやく，ドライバーズアイでウイニングランできたんです。あたしもう感動（うるうる）。

◆どうして“TeX”を「テフ」って読むのか理解できないんですけど。もしかして，ロシア語では？ う〜む，奥が深い。英語読みで「テックス」って読むのやっぱりダメなのでしょうか。

小宮山　博志(17)長野県

TeXブックを読みましょう。

◆最近あった私と友人の会話。
私：よくOh!Xにサイレントメビウスの話が載っているなあ。
友：この手の話って必ずレビアが出てくるんだよな。
私：個人的には女キャラより銃に脅されて焦るデューイ（レビアが製作した第五世代コンピュータの名前）が好きなんだけど。
友：オマエが女に縁がないのはカッコ悪いだけが理由じゃないんだな。
私：（大きなお世話だ）大平　浩貴(18)埼玉県

やっぱり僕は，レビアがいちばん好きですね。

◆Oh!X 1991年8月号の「編集室から」に載っていたOPMDに代わる新しいX68000用ミュージックドライバはどうなったんだ！ あれ以来音さたなし！ 開発が遅れてもいいから必ず発表してください。絶対買うぞ。大塚　博文(20)東京都

これが載る頃にはもう手にしているかな？

◆1万円を超えるソフトを出している某ソフトハウスのアンケートハガキにこんな質問があった。「1カ月に何本くらいソフトを購入しますか？」これで一部のソフトハウスではユーザー層を把握していないのがわかった。売り上げ計算をかなり多めに設定しているのではないかと思われる。　　河野　陽(18)大阪府

◆一部の悪のために多くの善が悪と思われてしまう。こんなことでX68000が潰されてしまったらたまらんよな。　　大山　将明(22)兵庫県

◆10月号の「X68000ゲームソフトのゆくえ」でX68000がひどいことをいわれているのを知りました。あまりにもひどすぎる。記事を読んで



◀大島　貴成　東京都
別に女の子でなくてじ〜さんでもハガキがおもしろければ載りますよ。一応12月号ですがちょっと気の早いクリスマスだね。

◀尾形　雅治　広島県
さりげなく4周年祝いのハガキがうれしいな。そう，知らぬ間に4年も過ぎていたんですね。これからも元気いっぱいがんばっていきましょう。

いて，よく山下章氏はすごいことをいった，と思いました。まだ，記憶に新しいウイルス事件のことでもそうですが，ウワサにはささいなことでもムチャクチャなものになるんですね。このことによって，Oh!Xを読んでいる正規ユーザーのソフトハウスに対する態度は以前以上に厳しいものになるでしょう。　水谷　国宏(17)滋賀県

◆10月号でいちばん目立ったのが特別レポートでした。LOGINにも似たようなものがありましたし，ほかの雑誌でもそうでしたね。私は今年に入って市販ソフトは３本しか買っていません。そうそう何本もソフトを買えるほど金持ちではないのです。よって，「買い支え」には協力できません。ソフトを買うことは慈善事業ではないのです。売れるソフト，そうでないソフトがあって当然。ソフトメーカーのために買い続けるのであれば，メーカーの前にユーザーのほうが潰れてしまいますよ。　加藤　雅浩(22)岡山県

◆不正コピー問題について私なりの考えを書いてみます。たとえ話になりますが，ファミコンのゲームでいちばん人気があって売れたと思われるのが「ドラゴンクエストIII」で，なんと350万本だといわれています。パソコンユーザーには考えもつかない本数でしょう。そして，ファミコン本体は1400万台以上出回っていることを考えれば，単純に計算して４人にひとりは買っていることになります。これをX68000の出荷台数を13万台として当てはめてみると約３万本になるでしょう。あれだけ大騒ぎして，しかもコピーの不可能なソフトでもこれなのだから，X68000で５～６万本も売れることなど，まず考えられないでしょう。実際問題はこれほど単純ではないでしょうけど。それからソフトの価格とコピーの関係は，全然とまではいわないがほとんど関連はないことと思います。なぜなら，たとえ１万円以上のものでも買うに値すると考えれば買うだろうし，千円のものでも借りる人は借りるはずだからです。　岡田　伸一(23)京都府

◆X68000のゲームの個々の販売数が伸び悩んでいるのは，やむをえないでしょう。ユーザー数はPC-9801よりもずっと少ないうえにゲームのクオリティが低い。ズー○のゲームはよく話題になりますが，私にとってはどれも起動してから５分ともたない。とてもじゃないがすべてのユーザーがこんなゲームについていけるとは思いません。マニア向けのゲームしか作っていないのに，現在より販売数を増やすのは無理でしょう。私はコピーできたとしてもする気になりません。　指中　芳夫(20)富山県

祝！
Oh!X
4周年!!
PON!
僕が購読し始めたのが「最後のOh!MZ」だったんですよね。
そうですか，もう4年になりますか。
おめでとうございます。これからもドラゴン精神を忘れずにがんばってください。

▲米山 一輝 大阪府
こちらも４周年祝いのハガキ。SD版の祝一平とホンニャアがとってもかわいい。そして，これからもよろしくね。

◆「X68000ゲームソフトのゆくえ」はなかなかみごとな分析結果でした。ただ，それほどゲームをしたいと思わない私は，もっとゲーム以外のものがマシンを支えていってほしいと思うのです。　岩崎　直明(23)千葉県

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★発足以来５年を超えるパソコンサークル「EXTRA」では，新規会員を募集します。主な活動は会員からの投稿を載せた会誌を発行しています。内容は簡単なお便りから，アルゴリズム講座，プログラミング関係，Q&A，売買など多岐にわたっています。また，一部の機種ではS-OSなどの打ち込みプログラムやMMLデータ，数は少ないけど会員の方の自作プログラム（ゲーム，ツール）の配布を行っています。特定機種を対象としていませんのでX68000からX1，MZ，PC-9801，PC-8801，MSXシリーズなど幅広く記事を載せていくつもりです。プログラムについては入会しなくても入手できますので興味を持った方は，62円切手を同封して当会に興味を持たれた理由を書いてお問い合わせください。折り返し案内書を送ります。〒811-42 福岡県遠賀郡岡垣町戸切794-3 筑紫 高宏(24)

★「Open Space」会員募集のお知らせです。「Open Space」では主にゲーム関係，プログラム技術の紹介を行っています。対象機種は特に問いません。会報は毎月１回，月の末日に発行しています。詳しいことは下記の住所までお問い合わせください。〒399-07 長野県塩尻市片丘10391 古旗 一浩(21)

★「OREGA」では，年８回の会誌発行を中心に幅広く活動しています。会報はプログラミング講座「はじめての3D」連載，ハードウェア講座「ぴーぷエレキ板」，テクニカル情報，ゲーム，パソコン通信情報，読書案内，エッセイ「OLからのひとこと」連載，SF，ボードゲーム，イラストなど盛りだくさんです。入会希望の方は，124円分（62円×２）の切手を同封のうえ，郵便番号，住所，氏名を明記して下記までお送りください。折り返し入会案内書を送ります。〒910 福井県福井市文京4-9-5 メゾン山本201 新海 敏之方「OREGA・入会希望Ｘ」係

売ります

★X1/X68000用カラーイメージスキャナ「CZ-8NS1」を70,000円で。X68000用カラーイメージユニット「CZ-6VT1」を20,000円で売ります。それぞれ箱，付属品，説明書すべてあり。連絡は官製ハガキでお願いします。〒520 滋賀県大津市におの浜２丁目2-5-519号 元田 善樹(16)

★インクジェットプリンタ「IO-730」を50,000円で。X68000用トランスピュータボード＋αを100,000円以上で売ります。トランスピュータボードは高く買ってくれる人優先です。連絡は官製ハガキでお願いします。〒815 福岡県福岡市南区那の川1-9-5 重藤 賢一(26)

買います

★X1用FM音源ボード「CZ-8BS1」を送料込み10,000円以下で買います。付属品，取扱説明書ありを希望します。連絡は往復ハガキにてお願いします。〒386-04 長野県小県郡丸子町下丸子1-5 金井 徳彦(31)

バックナンバー

★Oh!MZ1986年8，9月号，1987年5，11月号，Oh!X1988年１月号，1989年４月号をそれぞれ1,200円（送料込み）で買います。1987年11月号以外はS-OS“SWORD”の記事が完全であれば切り抜き汚れ可。連絡は官製ハガキでお願いします。〒300-32 茨城県つくば市佐470 川田 栄(18)

★C MAGAZINEのバックナンバー1991年５月号を1,500円で買います。傷や多少の汚れは別にかまいませんが，付録のディスク付に限ります。連絡は官製ハガキでお願いします。〒046 北海道余市郡余市町黒川町485番地 佐竹 寛(20)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々の意見を紹介しています。今月は10月号の内容に関するレポートです。

●特集「マシン語との邂逅」でいちばんよかったのは，「吾輩はX68000である」です。はい。本領発揮ってところですかねえ。もう無敵！ C言語云々いってどうなるんだろう〜って思ってたけど，息を吹き返したE.Tのようだ(なんのこっちゃ)。もう目からコンタクトが落ちました（嘘。コンタクトしてないもん)。これで漫画とか入ったら最強だと思うんですが，Oh!Xでは漫画で説明することはほとんどないんですね。いい漫画はいいですよ，うん。僕はI/O（1988年頃の奴だったかな）の漫画で，というか説明図で，Cの変数スコープやstaticのことを理解して感激したことがあります。“吾輩”のいいところはアセンブラすら超越して，マシンコードレベルで，それも擬人化して説明しているところ。イメージが摑みやすいぞ！

ただ，スタックについていわせてもらえば，なにか特殊なものではなくて，アドレスレジスタを効率よく使ってくれるマクロ命令だ，くらいの説明がほしい。スタックというと積むだの，出すだのという説明＋概念図というのが決まりのようだが，実態はアドレスレジスタを使ったデータ転送以外のなにものでもないのだから。なんか，あの決まりきったスタックの絵を見ると，魔法を使っているような気になります。どうも僕は実態がわからないとなんにもわからないたちで，VRAMだってRAMをコントローラが読み出して1ビットずつ同期させて送り出しているとわかるまで，なんでなんでとわめいていた。実態を伝える，というのを大事にしてください。

松村 知己(21) FM-7 石川県

●突然ですが，邂逅の意味。「思いがけず出会うこと」(角川書店「新字源」より)。私とマシン語との出会いは，まさに「邂逅」という感じでした。それはともかく，私はデバッガが好きです。あの重箱の隅をつつくような感覚がいいのです。なにかいえば（命令なり，なんなり実行すれば，の意)，必ずなにかが返ってくる。それが変なものでも，答えが返ってきてくれるのはうれしいことです。

いつぞやSX信州で遊んでいて，あのドクロマークのシステムエラーが出たことがありまして(2度か3度)，しかたないのでデバッガで追い掛けていくと（トレースかステップか，どちらかで)，「見ぃーつけたっ！」とばかりにバスエラーの素（だって，“by bus error”でしょ？)。思わずほくそ笑んでしまいました（アブナイ女だ)。結局しばらく悩んで私の負け。で，信州はというと，ちゃんと動くよう……です。まあ，こんなことばかりじゃなくて，自作プログラムの虫が取れたこともありますよ。アセンブルのときにWarning出されてても気がつかないの。

安井 百合江(17) X68000 PRO 愛知県

●私のX68000は優れたゲーム機であると同時に，手軽な“楽器”でもあります。私がX68000を選んだ理由のひとつは，その強力なAV機能でした。ろくな音楽的知識もOPMの技術もない私のことですから，身近にある楽譜を内蔵の音色で鳴らす程度なのですが，それでもそれなりのものができるのは，やはりOPMの性能によるところが大だと思います。いままで，時として自作の音色で自作の曲を作りたいと思うことはあっても，そんなのは夢にすぎず，偶然にできた，“曲”ともいえぬ代物を@39や@15等でろくな伴奏もつけず鳴らすだけだったのです（いまもそうですが)。自分でも少しは音楽について知ろうと入門書を探しているのですが，なかなかいい本が見つかりません。どうも音楽というのは“暗黙の了解”が多く，つかみにくいもののようです。まあ，MMLさえ知っていれば，とりあえず曲は鳴らせるわけですが，やはりそれだけというのも，もったいない話であります。そんななかで，10月号から始まった「Creative Computer Music入門」に期待しています。

宍戸 輝光(17) X68000 PRO 東京都

●「GSフォーマットを斬る」を読んでいて，昔のFM，PC間でのフロッピーのやりとりのことを思い出してしまいました。FDD（というか，FDC)のほうでは読み込めるはずなのですが，実は読み取れないという困りものでした。両者間の変換用ツールが昔のI/Oなんかに載っていたのを覚えています。現在のように一応MS-DOSに規格が定まっていて，X68000のように異なるCPUのマシンでもファイルを見ることができるのはうれしいことです。

音楽データのほうも現在過渡期にあるのでしょうが，フォーマットの統一によるメリットは理解できても，それに従うのを潔しとしない音源メーカーも多いと思います。各メーカーの考え方がありますから，GSフォーマットもローランド1社が提唱しているだけというのはかなり危ないですね。ローランドはこの世界では大手でしょうが，ほかのメーカーがあっさり従うとも思えません。それにまだ，フォーマット自体が確定していないようですし，統一はまだ先のことでしょう。

中村 健(21) X68000 ACE-HD,MSX2+ 埼玉県

ごめんなさいのコーナー

10月号 Oh！X LIVE in '90

SPANISH BLUEの音色設定プログラムが抜けていました。以下に掲載します。

```
B000 FE 03 C2 60 20 F5 E5 78 : 95
B008 FE 03 C2 DC B0 1A 13 B7 : 33
B010 CA 6F 20 FE 29 D2 6F 20 : E1
B018 3D 26 00 6F 29 29 44 4D : B5
B020 29 29 29 09 01 90 B1 09 : CF
B028 D5 DD E1 DD 5E 00 DD 56 : 01
B030 01 D5 DD E1 DD 7E 12 0F : 10
B038 0F DD 86 00 77 23 DD 7E : 67
B040 0E 07 07 07 07 DD 86 10 : 9D
B048 77 23 06 06 DD E5 10 FC : 74
B050 11 16 00 06 04 DD 7E 26 : B2
B058 07 07 07 07 DD 86 24 77 : 1A
B060 23 DD 19 10 F0 DD E1 06 : DD
B068 04 DD 7E 20 77 23 DD 19 : 0F
B070 10 F7 DD E1 06 04 DD 7E : 2A
B078 22 0F 0F DD 86 16 77 23 : 53
-------------------------------------
SUM: 07 5A A8 78 8D 7A 72 F1 D8B5

B080 DD 19 10 F2 DD E1 06 04 : C0
B088 DD 7E 2A 0F DD 86 18 77 : 86
B090 23 DD 19 10 F3 DD E1 06 : E0
B098 04 DD 7E 28 0F 0F DD 86 : 08
B0A0 1A 77 23 DD 19 10 F2 DD : 89
B0A8 E1 06 04 DD 7E 1E 07 07 : 72
B0B0 07 07 DD 86 1C 77 23 DD : 04
B0B8 19 10 F0 DD E1 06 05 36 : 18
B0C0 00 23 10 FB DD 7E 08 77 : 08
B0C8 23 DD 7E 0A F6 80 77 23 : 98
B0D0 DD 7E 0C 77 23 DD 7E 04 : 60
B0D8 77 23 36 00 E1 F1 C9    : 6B
-------------------------------------
SUM: 73 86 95 D2 27 CA C3 9C DB20
```

バグに関するお問い合わせは
☎03(5488)1311(直通)
月〜金曜日16：00〜18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

音楽を聴く 音楽を奏でる 音楽を創造する

▼CDやテープで音楽を聴く人は多い，というか，ほとんどの人がなんらかのかたちで音楽を聴いているといえます。それぐらい音楽を聴くという行為はごく自然なものという認識があります。しかし一方，ほとんどの人が楽器を演奏している，あるいは作曲しているとは，決していえないと思われます。

誰もが小学校では音楽の時間に楽器を使ったことがあるのに，ずっと続けるということにはなっていない。もちろん音楽が大きらいという人もいるにはいるでしょう。しかし，音楽を聴いているという人は音楽が好きなはずです。なぜ，そこから創造の側へと踏み出さないことが多いのでしょうか。

まず，わざわざ楽器を手に入れるのが面倒臭い。それに自分には才能がないから，聴くだけでいいという理由も多いのではないでしょうか。

しかし，我々の前には楽器も，才能を手助けしてくれる有能な助手も存在しています。そう，パソコンです。

最近のパソコンにはわりと高機能な音源が載っていたり，MIDIで音源モジュールをコントロールしたりできます。また，ソフトの面でもノウハウが蓄積されつつあり，楽器と紙と鉛筆だけで創作を行うよりは楽になっているはずです。

今月号の特集のようなかたちで，いろいろな音が簡単に作れたり，演奏方法が身近になれば，目の前にある箱はすでにひとつの楽器となるでしょう。そして，そのあとはあなた次第で可能性は無限に膨らむのではないでしょうか。

▼今月のOh!Xの発売とほぼ同時に，Z-MUSICシステムのムックが発売されました。ディスク3枚付きとなっていまして，価格は2,300円（税込）です。Oh!Xでは引き続きこのような形態でメディアを供給する予定です。出さなければならないものはたくさんあるのですが，とりあえず，次はSX-WINDOW関連になりそうです。ご期待ください。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒108 東京都港区高輪2-19-13 NS高輪ビル
ソフトバンク出版部
Oh!X「㋢㋼㋮㋯」係

SHIFT・BREAK

▶TVが壊れた。45度チョップを浴びせてみたが直らないので，ゲームギアとTVチューナーパックを衝動買いする。机の上に置けて原稿描きながらでもTVが見れるし，画質もなかなかよろしい。おまけにゲームもできる。こんないい機械があまり売れてないらしい。でも，そこがまたたまらなくSHARP党の脳髄をくすぐるのでした。ああ，悲しい性（笑）。（哲）

▶10月になって深夜番組がガラっと変わってしまい生活のリズムが取れなくなった。特に「ビデオの女王様II」が終わってしまった影響は大きい。しかたがないので「アジア台風ショー」を見てから「おちゃめなふたご」のビデオを消化することにしてみたが，このアニメも終わってしまうので，テレビ番組に飢えている今日この頃であった。（八）

▶11月号のアフターレビュー（遥かなるオーガスタだった）を見た。みんな頑張っているなと，同じオーガスタプレイヤーの私としては嬉しかった。ふっふっふっ，私のスコアは自己ベスト－13，ロングパットは96フィート（自慢！），チップインが200ヤードだ。奈良県の川脇君，今日のところは引き分けだね。（就職の決まった毛）

▶アリスソフトから送られてきた「ランスIII」を遊んだ。相変わらずの主人公ランスの畜生振りを見て感激。バグが結構あるのには困惑したが，内容的には，I，IIから段々とパワーアップしていることを実感。エンディングでIで出てきたコンクリート詰めの親父が再登場したのには大爆笑。僕も大人になったら「ランス」みたいになるんだ。（子供の善）

▶私の母は，知るかぎり最強の晴れ女である。台風21・22号の同時接近の真っ最中だったにもかかわらず，10/10の1周忌は雨が降らなかった。前日も翌日もどしゃ降りだったのに。そういえば，火葬の日も家の中にいる時は降っていた雨が，移動するときにはピタリと止んでいた。父も私も，いまだかつて墓参りのときに雨に降られたためしはない。（S.K.）

▶紀元前3000年から人々は自らの手で自然を破壊していたらしい。農耕を始めた時点で未来は定められた。人間の起源はアフリカ，という説が有力だ。そこからアメリカ大陸やオーストラリアにまで人類は広がった。4大文明よりもずっと昔の話。アメリカ先住民が東洋系の顔をしているのは，アジア→アラスカ→アメリカ大陸と渡っていったからだ。（K）

▶新宿にある新都庁舎を見にいった。ガウディか何かを思わせる異様な外観は，本当にこれがお役所かと疑いたくなる。ここの目玉は45階（地上200メートル！）にある展望室だ。「入場無料」ということばにつられて上ったが，窓に近づくとなぜか気分が悪くなって，景色の素晴らしさを楽しむどころではなかった。これが高所恐怖症なのか。（KO）

▶祝！ 高河ゆん完全復活。実にうれしい。初めて聞いたときには半信半疑の状態だったけど，最近予告どおり誌面に掲載されるようになってから，ようやく信じることができた。「源氏」も連載を再開するし，そのほかの連載も復活していくようだ。ファンのひとりとして素直によろこんでしまう。とりあえず，よかったよかった。（J）

▶今月のmicroOdesseyは佐渡島漫遊記などと銘打って，佐渡金山の中の人夫ロボットの話でも書こうかなと思っていたのだが，大幅に変わってしまった。最初は“SOFTWARE INFORMATION”向けに軽くいこうと書き始めたら，指が勝手に動き回って，20分後には次ページのようになったのだ。これは机上の“テレビ！”の呪い，いや効力なのだろうか。（A）

▶邦楽漬けの反動か，仕事中突然BON JOVIが聴きたくなった。が，当然編集部にゃそんな趣味のヤツはいない。3歩譲ってVAN HALENかSKID ROWでもいい，と社内中探しまくったがやっぱりあるわきゃない。1万歩譲ってX（結局邦楽だ）を聴いていたら案の定眠ってしまった。気がつきゃもう朝，仕事は終わらず。うー，自己嫌悪。（E.O.）

▶久々に特集をやるぞ，と思っていたらMookのほうでも手一杯（MookはMagazineとBookの合成語です）。ぶっつけ本番のDTP，飛ぶMacII，落ちるプリンタ，遅い重い遅い！ マルチタスクだから印刷中に編集も……なんてのは両手でロシアンルーレットをやるようなものだ。第2弾はどうなることやら。（さすがにバテ気味のU）

▶10月20日はX68000芸術祭の審査で名古屋へ。徹夜明けでボーッとしながらも新幹線にDynaBookを持ち込み，富士山を横目に原稿書き。シャープさんのご好意により，帰りのための充電も万全だった。ところが，その日の夜は鈴鹿帰りのF1小僧でホームは溢れんばかり。結局DynaBookも重いだけになってしまった。こんどはPowerBookにしようかな。（T）

microOdyssey

プロサッカー68大会奮戦記

イマジニアの新作,「プロサッカー68」のゲーム大会がパソコン雑誌編集部対抗で行われた。参加したのは,テクノポリス,ポプコム,コンプティーク,マイコンBASICマガジン,マイコン,ログイン,そして,我がOh!Xの全7誌。ほかの雑誌では編集者にライター,カメラマンなどと,2,3人の複合パーティーを組んで戦いに備えていたが,勇気あるOh!Xでは編集者の私ただひとりが会場へと乗り込むことになった。単に都合がつかなかっただけという噂もある。

というわけで,単身で会場に乗り込んだ私は用意された席に着いた。独り身は暇なもので,しばらくまどろんだのち,ふと顔を上げると,そこには黒装束を身に着けた忍者の姿があった。「なんじゃ,こいつは〜」と大声を出しそうになるのを抑えつつ,よくその忍者を見ると……,もうおわかりだろう。説明は省くが,わからない人はログインを読むといい。

と,そんなこんなで参加者が揃い,いよいよ大会が始まることになった。まずはくじびき。大会はトーナメント方式で行われるので,その組み合わせを決めるためだ。参加したのは7誌だから1誌はシードとなるのだが,おいしいところはテクノポリスにさらわれた。

第1回戦

相手は忍者,つまりログインだった。試合前の握手をするときに,毒のついた手裏剣でも握らされるかと思ったが,それは取り越し苦労だったようだ。しかし,相手は忍者,とんでもない術を繰り広げるかもしれない。そして,その予想は思わぬかたちで現実化した。自殺点。なんてことをするのだ。リラックスしてしまったじゃないか,おい。というわけで,そのあと1点追加し,2−0で勝ってしまった。

第2回戦

ベーマガとの内部抗争の結果,生き残ったマイコンとの戦い。なんとなくプレイしていたら,なんとなく勝ってしまった。終わり。

決勝戦

ポプコム,そしてシードのテクノポリスに勝って,這い上がってきたコンプティークが相手。ここまで来たのだから,たぶん強いに違いない。こちらは「プロサッカー68」は2回ぐらいしかプレイしてなく,ここまで勝ってきたのが不思議なくらい。あっさり負けるのではないだろうか,でもここまで来たら優勝賞品の液晶テレビを持って帰りたい。そんな2つの思いが交錯するなか,火蓋は切られた。

前半はいい勝負だった。とはいっても,両者がうまかったのではなく,お互いうまくボールをコントロールできないまま試合が進んでいたといったほうが正しいだろう。さてはここにいる全員が練習不足だなという結論に達した。そして,ハーフタイムに入る直前,シュートが決まった。思わず小さくガッツポーズが出て,「テレビ!」と心の中で叫ぶ。そのあと相手側は選手交代をしたが,テレビに対する執念の火の手はおさまらず,結局3−0で勝った。

表彰式ではさすがに頬が緩み,だらしない顔のまま賞品の4インチ液晶カラーテレビ(もちろんシャープ製)を受け取った。そして,外に出たあと公衆電話を探し,編集部へ勝利報告の電話を入れたのである。 (A)

1992年1月号12月18日(水)発売

特集 SX-WINDOWの未来

・ウィンドウシステム比較/システムの可能性を探る

Oh!X LIVE in '92 DRAGON SABER STAGE4

MAGIC用迷路ゲーム

新製品紹介

音源モジュール CM-300/500

Press Conductor PRO-68K

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートB1	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3200)9185
	渋谷	大盛堂書店	03(3463)0511
	池袋	リブロ池袋店	03(3981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(3981)0111
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ,必要事項を明記のうえ,郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので,大切に保管してください。なお,すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は,上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店,日本IPS(株)にお申し込みください。なお,購読料金は郵送方法,地域によって異なりますので,下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700



12月号

■1991年12月1日発行 定価600円(本体583円)
■発行人 孫 正義
■編集人 橋本五郎
■発売元 ソフトバンク株式会社
■出版事業部 〒108 東京都港区高輪2-19-13 NS高輪ビル
Oh!X編集部 ☎03(5488)1309
出版営業部 ☎03(5488)1360 FAX 03(5488)1364
広告営業部 ☎03(5488)1365
■印 刷 凸版印刷株式会社



満開の電子(でんこ)ちゃん

作・え 岡村祭











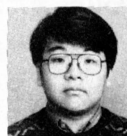








購読方法：定期購読もしくはソフトベンダー武尊(タケル)でお買い求めいただけます。
★定期購読の場合＝定期購読料６ヶ月分6,000円（送料サービス、消費税込）を、現金書留または郵便振替で下記の宛先へお送り下さい。
　現金書留の場合：〒171　東京都豊島区要町1—19—3　いさみビル4F　満開製作所
　郵便振替の場合：東京5—362847　満開製作所
●御注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
●新たに購読を開始される方は、「新規」とご明記下さい。
●製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。
★武尊でお求めの場合＝１部につき1,200円（消費税込）です。
●定期購読版と内容が一部異なる場合があります。ご了承下さい。
●お問い合わせ先　TEL（03）3554—9282（月～金　午前11時～午後６時）
（なお、定期購読版のバックナンバーについては定期購読者の方のみご注文を承ります）



石倉裕之（神奈川県）

「満開製作所」の「電脳倶楽部」。その存在を知ったのは3年前、Oh!Xの広告でした。X68000を手に入れてからも「あやしい」ただそれだけの理由で購読していませんでした。しかしある日、ふとした思いつきで半年の契約をしてしまったのです。ところがこれがなかなかこない！　しまったかあ！と思っていたところに、やってきました28号。中にはぎっしりつまったツール、グラフィック等…ハマりました。「満開製作所」「電脳倶楽部」。うーん良い響きではありませんか！（やっぱりあやしいけど）

年末度、大特価セール開催中！ X68000 XVI

エクシヴィ

16Mhzの、MC68000搭載
体感速度 約2倍!!

マルチウィンドゥシステム
疑似マルチタスク処理
本格的なGUI環境を実現する
「SX WINDOW Ver 1.1」
SCSI I/F 標準装備
本体内蔵 拡張メモリースロット採用
最大8MBメモリー内蔵可能(12MBまで拡張可)

エクシヴィ 快走!!

直接ご来店頂けない場合は、通信販売もご利用頂けます。
お近くの「OAシステムプラザ」迄、お電話にてお申し込みください。

お電話をお待ちしております。
お近くの「OAシステムプラザ」へ
多数取り扱っております!!
その他 各種周辺機器、中古品 等

¥OA特価販売中！ ※クレジット金額は均等払いの目安です。

X68000 XVI

メインメモリ2MB標準実装、80MBハードディスク内蔵可能
16MHzクロック、世界標準SCSI I/F内蔵、縦型モデル

CZ634CTN 標準小売価格 ¥368,000
CZ606DTN 標準小売価格 ¥79,800
標準価格合計 ¥447,800

OA特価販売中！

お支払回数	12回	24回	36回
毎月お支払金額	¥30,900	¥16,300	¥11,500

X68000 XVI

メインメモリ2MB標準実装、80MBハードディスク内蔵可能
16MHzクロック、世界標準SCSI I/F内蔵、縦型モデル

CZ634CTN 標準小売価格 ¥368,000
CZ614DTN 標準小売価格 ¥135,000
標準価格合計 ¥503,000

OA特価販売中！

お支払回数	12回	24回	36回
毎月お支払金額	¥34,700	¥18,300	¥12,900

X68000 XVI-HD

メインメモリ2MB標準実装、80MBハードディスク内蔵
16MHzクロック、世界標準SCSI I/F内蔵、縦型モデル

CZ644CTN 標準小売価格 ¥518,000
CZ614DTN 標準小売価格 ¥135,000
標準価格合計 ¥653,000

OA特価販売中！

お支払回数	12回	24回	36回
毎月お支払金額	¥45,100	¥23,800	¥16,800

X68000 PROⅡ

メインメモリ1MB標準実装、40MBハードディスク内蔵可能
世界標準SCSI I/F内蔵、縦型モデル

CZ653CBK 標準小売価格 ¥285,000
CZ606DBK 標準小売価格 ¥79,800
標準価格合計 ¥364,800

¥218,000

お支払回数	12回	24回	36回
毎月お支払金額	¥20,100	¥10,600	¥7,500

X68000 SUPER

メインメモリ2MB標準実装、80MBハードディスク内蔵可能
世界標準SCSI I/F内蔵、縦型モデル

CZ604CTN 標準小売価格 ¥348,000
CZ606DTN 標準小売価格 ¥79,800
標準価格合計 ¥427,800

¥258,000

お支払回数	12回	24回	36回
毎月お支払金額	¥23,800	¥12,500	¥8,800

X68000 SUPER-HD

メインメモリ2MB標準実装、80MBハードディスク内蔵
世界標準SCSI I/F内蔵、縦型モデル

CZ623CTN 標準小売価格 ¥498,000
CZ606DTN 標準小売価格 ¥79,800
標準価格合計 ¥577,800

¥318,000

お支払回数	12回	24回	36回
毎月お支払金額	¥29,300	¥15,400	¥10,900

※実装方法など各支店の「PRO STUFF」までお気軽にご相談ください!!

IOデータ機器製 純正互換増設RAMボード

PIO6BE1A	(1MB内部増設RAMボード)	➡ ¥17,800
PIO6BE2-2M	(2MB増設RAMボード)	➡ ¥35,800
PIO6BE4-4M	(4MB増設RAMボード)	➡ ¥61,800
SH−6BE1-1M	(CZ600C専用1MB増設RAMボード)	➡ ¥28,000
SH−6BG1	(GPIB I/Fボード)	➡ ¥44,800
SH−6BF1	(RS232C 2チャンネル増設 I/Fボード)	➡ ¥37,400
SH−6BN1	(イメージスキャナー用 パラレルI/Fボード)	➡ ¥22,400
SH−6BU1	(ユニバーサルI/Oボード)	➡ ¥29,800

HAL研究所 ファインスキャナー256

X68000専用ハンディイメージスキャナー
グレイスケール(256階調)対応
読み取り幅105mm 解像度 100/200dpi

標準価格 ¥39,800
¥31,800

SHARP純正 拡張インターフェースボード

CZ−6BE1	(CZ600C専用1MB増設RAMボード)		➡ ¥28,000
CZ−6BE1B	(1MB内部増設RAMボード)		➡ ¥22,400
CZ−6BP1	(数値演算プロセッサーボード)		➡ ¥63,800
CZ−6BS1	(SCSI I/Fボード)		➡ ¥23,800
CZ−6BF1	(RS232C 2チャンネル増設 I/Fボード)		➡ ¥39,800
CZ−6BM1	(MIDI I/Fボード)		➡ ¥22,400
CZ−6EB1	(拡張I/Oボックス)		➡ ¥69,800
CZ−6BV1	(ビデオボード)		➡ ¥16,800
CZ−6BN1	(GPIB I/Fボード)		➡ ¥23,800
XVIシリーズ専用タイプ			
CZ−6BE2A	(XVI専用内蔵2MB増設RAMボード)	¥59,800	➡ ¥47,800
CZ−6BE2B	(CZ6BE2A増設用 2MBRAM)	¥54,800	➡ ¥43,800
CZ−6BP2	(XVI専用内蔵数値演算プロセッサー)	¥45,800	➡ ¥35,800

ソフトウェア

多機能プリントエディター
Print Shop Ver.2.0
CZ−265HS
標準価格 ¥19,800
¥17,800

遂に発売／多機能ワープロソフト
マルチワード
CZ−225BS
標準価格 ¥32,000
¥28,000

本格的DTPソフト
Press Conductor
CZ−266BS
御予約受付中！

X68000用ハードディスク

80MB SASI/SCSI両対応
TX-80 定価 ¥108,000
¥88,000

130MB SCSI方式
TX-130 定価 ¥138,000
¥108,000

180MB SCSI方式
TX-180 定価 ¥185,000
¥148,000

ROLAND MIDI音源モジュール SC-55「サウンドキャンバス」

MIDI楽器の新しい規格「GS音源」
16パート、リズム音源内蔵と
一台で本格的なアンサンブルが可能
MT32、CM32L上位コンパチ機種です

ROLAND SC-55 ¥69,000
システムサコム SX68M ¥19,800
標準価格合計 ¥88,800
¥74,000

ROLAND CM-32L ¥69,000
LAシンセ8パート リズム音源1パート
9パート同時発音可能
システムサコム SX-68M ¥19,800
標準価格合計 ¥88,800
¥74,000

ROLAND CM-64 ¥129,000
LAシンセ8パート リズム音源1パート
PCM音源6パート 15パート同時発音可能
システムサコム SX68M ¥19,800
標準価格合計 ¥148,800
¥125,000

銀行振込

各店舗に御予約、ご注文いただきましたら、最寄の銀行から当社指定銀行口座へ『電信扱』にてお振り込み下さい。手数料はお客様負担になります。

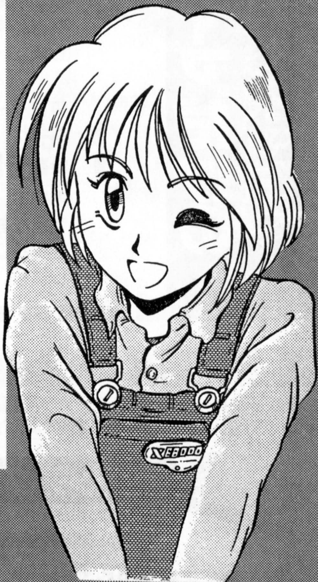
代金引き替え配送

お電話で商品の注文が出来ます。お客様宅へ配達時、商品と引き替えにお代金をお支払いいただきます。商品代金の他に手数料かかかります。

クレジット

お電話にてお申込みいただきましたら折り返し弊社より専用申込用紙をお送りいたします。必要事項記入の上ご返送下さい。

いずれも商品在庫をご確認の上お申し込みください。
※表示価格には消費税は含まれておりません。



札幌店	011-210-8812	大須店	052-265-1650
仙台店	022-268-5541	京都店	075-344-0347
東京店	03-3255-9188	大阪店	06-632-4233
横浜店	045-314-6634	大阪日本橋店	06-646-3169
浜松店	053-458-3755	岡山店	0862-21-4133
名古屋店	052-332-5233	広島店	082-240-9669
名古屋アメ横店	052-264-9715	福岡店	092-714-0030
アメ横2F店	052-262-6909	福岡ユーテク店	092-733-8931

札幌から福岡まで全国をつなぐ
X68000 PROSHOP

(株)OAシステムプラザ

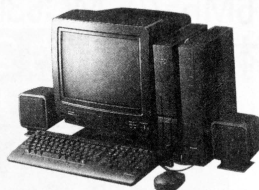
本社 愛知県名古屋市中区大井町3-20
OAビル

パソコンショップ

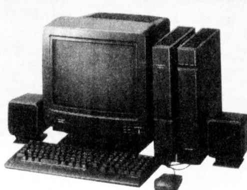
R&R

X68000 年末特別セール

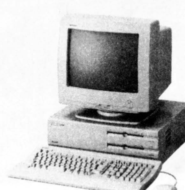
X68000 XVI



X68000 SUPER



X68000 PROII



XVI基本セット

CZ-634C-TN	定価	¥368,000
CZ-606D-TN	定価	¥ 79,800
合　計		¥447,800
R&R提供価格		**¥3?0,000**

大特価セール!!

CZ-623C-TN	定価	¥498,000
CZ-606D-TN	定価	¥ 79,800
合　計		¥577,800
R&R提供価格		**¥328,000**
CZ-604C-TN	定価	¥348,000
CZ-606D-TN	定価	¥ 79,800
合　計		¥427,800
R&R提供価格		**¥268,000**

コンピュータグラフィックセット

CZ-634C-TN	定価	¥368,000
CZ-606D-TN	定価	¥ 79,800
CZ-8PC5	定価	¥ 96,800
Z'S STAFF PRO-68K V2.0	定価	¥ 58,000
合　計		¥602,600
R&R提供価格		**¥448,000**

パソコン通信セット

CZ-634C-TN	定価	¥368,000
CZ-606D-TN	定価	¥ 79,800
MD24FB5V	定価	¥ 39,800
たーみのる2	定価	¥ 17,800
合　計		¥505,400
R&R提供価格		**¥378,000**

コンピュータミュージックセット

CZ-634C-TN	定価	¥368,000
CZ-606D-TN	定価	¥ 79,800
SX-68M	定価	¥ 19,800
CM-64	定価	¥129,000
MA-12C(2台)	定価	¥ 28,000
Mu-1 SUPER	定価	¥ 39,800
合　計		¥664,400
R&R提供価格		**¥498,000**

コンピュータミュージック

SC-55	¥ 69,000
SX-68M	¥ 19,800
Mu-1 SUPER	¥ 39,800
合　計	¥128,600
R&R提供価格	**¥102,000**

周辺機器

	定価	R&R提供価格
▼プリンタ		
IO-735XB	¥248,000	**¥169,000**
▼増設メモリー		
CZ-6BE1	¥ 35,000	**¥ 27,500**
PIO-6BE1A	¥ 25,000	**¥ 19,500**
PIO-6BE4	¥ 88,000	**¥ 69,000**
▼その他のオプション		
CZ-6BS1	¥ 29,800	**¥ 23,800**

その他セット、周辺機器も取り扱っておりますので、お気軽にお問い合わせください。

※掲載商品の価格は、全て消費税別です。

R&Rメディアだけの年末4大特典!

特典1 特製Tシャツプレゼント!!

X68000版アルシャークを特製オリジナルTシャツ付きで**7,840円**(送料、消費税サービス)で販売します。

特典2 メンバーズカードを発行

期間中R&Rメディアで商品を買うと、「パソコンソフトが店頭価格よりさらに**5%OFF**」など、特典の付いたメンバーズカードを発行します。

特典3 特にゲームユーザーは注目!木村明広先生(ライトスタッフ所属)の描き下ろし特製テレホンカードをプレゼント!

期間中、店頭または通信販売で15,000円以上商品をお買い上げのお客様に木村明広先生描き下ろしの特製テレホンカードをプレゼントします。

特典4 ソフト通信販売価格が最大25%OFF!!

消費税はサービスになります。
(ただし、エニックス、ポニーキャニオン、ビクター、教育ソフトは除く)
他社にはない新システムを採用/お客様がお求めになる商品の定価の合計によって下記の通りサービス(割引率)が変わります。もちろん、他の特典も併せてご利用になります。

▼サービス内容　(消費税はサービス。( )内はメンバーズカードをお持ちの場合)

定価合計	(メンバーズカード)	割引	送料
定価合計が5千円未満		15%OFF	送料300円
〃 5千円以上1万円未満	(5千円未満)	20%OFF	送料500円
〃 1万円以上1万5千円未満	(1万円未満)	20%OFF	送料サービス
〃 1万5千円以上2万円未満	(1万5千円未満)	23%OFF	送料サービス
〃 2万円以上	(1万5千円以上)	25%OFF	送料サービス

●例:商品の定価合計が12,800円の場合10,240円(20%OFF、送料、消費税サービス)になります。メンバーズカードをお持ちですと9,856円(23%OFF、送料、消費税サービス)になります。

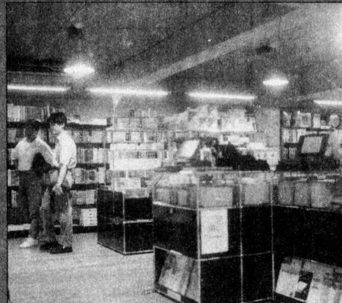
通信販売でのお申し込み方法

■商品は電話またはファックス(お客様の電話番号をお忘れなく)でご注文下さい。■お支払いは銀行振込でお願いします。入金確認後の発送となります。ソフトに関しては現金書留も可能です。ローンも取扱っています。■表示されている金額で特に記載が無い商品は送料・消費税は含まれておりません。■掲載以外にも各社商品を扱っておりますので、お気軽にご相談下さい。
■振込先:富士銀行 西大井支店 (普)1358191 アール・アンド・アール・メディア(株)

各種教室及びソフト体験コーナー開講中!!
詳しくはR&Rまでお気軽にお尋ね下さい。



●取扱い商品 NEC・富士通・エプソン・シャープ(メーカー保証付) ソフト、各種サプライ用品



パソコンショップ
R&Rメディア
☎03-3777-7335
〒140 東京都品川区西大井6-10-10 品川RSビル
営業時間/11:00～20:00(火曜日定休日)

年末謝恩セール!! (91.12.15迄)

全国送料 無料サービス実施中 (1万円以上お買い上げの方)

AIBIT アイビット電子株式会社

X68000XVI PROⅡセット

CZ-653C + CZ-606D

限定 **¥218,000**

XVI
CZ-634CTN

+CZ-606D	¥320,000
+CZ-604D	¥330,000
+CZ-612DGY	¥340,000
+CZ-607D	¥335,000
+CZ-614D	¥360,000

XVI HD
CZ-644CTN

+CZ-606D	¥430,000
+CZ-604D	¥440,000
+CZ-612DGY	¥450,000
+CZ-607D	¥445,000
+CZ-614D	¥470,000

SUPER
CZ-604CTN

+CZ-606D	¥268,000
+CZ-604D	¥278,000
+CZ-612DGY	¥288,000
+CZ-607D	¥283,000
+CZ-614D	¥298,000

SUPER HD
CZ-623CTN

+CZ-606D	¥315,000
+CZ-604D	¥325,000
+CZ-612DGY	¥335,000
+CZ-607D	¥330,000
+CZ-614D	¥345,000

EXPERT Ⅱ
CZ-603C

+CZ-606D	¥278,000
+CZ-604D	¥288,000
+CZ-612DGY	¥298,000
+CZ-607D	¥293,000
+CZ-614D	¥318,000

EXPERT Ⅱ
CZ-603C(内蔵40BMHD)

+CZ-606D	¥338,000
+CZ-604D	¥358,000
+CZ-612DGY	¥368,000
+CZ-607D	¥363,000
+CZ-614D	¥388,000

富士通電脳最前線'91

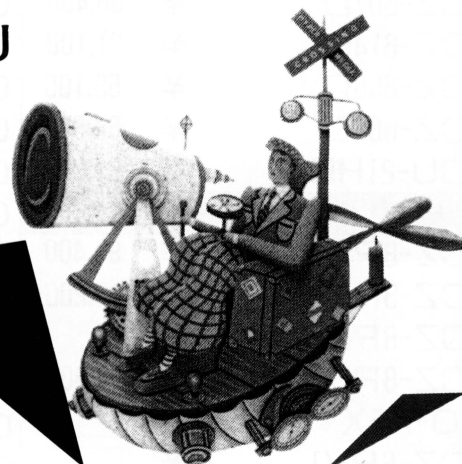
ハイパーステーション

12/6(金)7(土)8(日)

in東京ドーム

当店は電脳最前線'91
ハイパーステーションに
協賛しています。

FUJITSU



入場券を差上げます。ハガキにて御応募下さい。

X68000
オリジナルX68000セットロゴシール(大・小)とソフト2本プレゼント(12/15まで)

ドットマトリクス漢字プリンタ(136桁)
CZ-8PK10
標準価格 ¥97,800
特価

300/1200BPS全2重通信、ボイスメール対応、プライベートアンサーシステム構築も可能なハイブリッドモデムホン
MZ-1X30
標準価格 ¥98,000
特価 ¥19,800

電子手帳データーを自由にカッティング
MC-300
定価 ¥580,000
資料請求して下さい。

X68000 3.5インチフロッピーディスクユニット
OS-9/X68000用デバイス、ディスクプリンタ付HUMAN68K動作可
X6835-2F
標準価格 ¥80,000
特価

24ピン漢字プリンタ(80桁)
CZ-8PK7
標準価格 ¥122,000
特価 ¥59,800

カラーイメージスキャナ
232Cケーブル、スキャナツールソフト付
JX-220X 標準価格¥168,000
特価 ¥134,500

漢字水平プリンタ
MZ-1P27
標準価格 ¥268,000
特価 ¥98,000

HXD 040 23ms X68000
外付けハードディスク
標準価格¥118,000 特価 ¥75,000
HXD 140 X68000 内蔵用
40Mハードディスク
標準価格¥98,000 特価¥75,000
HXD 140は602C, 603C, 652C, 653Cの内蔵用

※富士通、NEC、シャープ周辺機器(拡張機器全機種、プリンター他)も常時取り扱っております。 〈全商品新品完全保証付〉 シャープ、カシオポケコン全機種取扱い。カタログ、価格表ご請求には、72円切手を添えてお願い致します。

通信販売のお問い合せ、御注文は

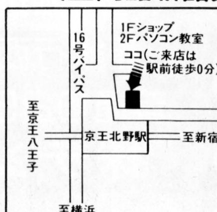
0426-45-3001(本店)

FAX.0426-44-6002

●営業時間/10:00~19:00●電話受付/20:00迄可●定休日/水曜日

SHARP SUPER EXE SHOP

アイビット電子株式会社 〒192 東京都八王子市北野町560-5



上記の広告商品はすべて店頭販売もしております。

全国通信販売 北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 (普)1752505

●本誌発売時には上記価格よりさらにお求めやすい価格に変更されている場合があります。●この広告の商品にはすべて送料・消費税は含まれておりません。

X SHARP PROSHOP 0286 22-9811 BASIC HOUSE

オリジナル SCSI HDD内蔵XVIセット

大好評につきさらに値下げ!



CZ-614D-TNセット

100M内蔵XVI ¥506,000
本体価格 ¥398,000
200M内蔵XVI ¥604,000
本体価格 ¥498,000

※他のディスプレイとのセット価格も格安!ぜひ電話でご確認下さい。
※電源投入後、約15秒で1度リセットの必要があります。

SUPER/BASIC HOUSE/特別セット

オリジナルハードディスクを内蔵
純正のHDD内蔵型よりも安い! 超高速12msee!!

決算大処分

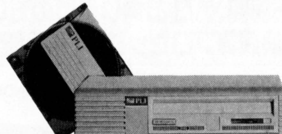
CZ-614D-TNセット

100M内蔵 SUPER ¥380,000
200M内蔵 SUPER ¥460,000

※他のディスプレイとのセット価格も格安!ぜひ電話でご確認下さい。
※電源投入後、約15秒で1度リセットの必要があります。

Infinitry 40 turbo -X68-

メディアの着脱が自在
scsi仕様ハードディスク
メディア1枚あたり42Mバイト
X68000用scsiケーブル、ターミネータ付属



OH!X 特別特価

メディア2枚サービス ¥148,000
メディア2枚&CZ-6BS1 ¥170,000

増設メモリ&コプロセッサボード

KGB-X68PRK IIシリーズ
購入後のグレードアップも出来ます。

2M実装/コプロ別売り	PRK II-02	¥55,000
4M実装/コプロ別売り	PRK II-04	¥90,000
6M実装/コプロ別売り	PRK II-06	¥125,000
8M実装/コプロ別売り	PRK II-08	¥160,000
2M実装/コプロ付属	PRK II-12	¥85,000
4M実装/コプロ付属	PRK II-14	¥120,000
6M実装/コプロ付属	PRK II-16	¥155,000
8M実装/コプロ付属	PRK II-18	¥190,000

旧PRK処分特価

PRK IIの新発売に伴い、旧PRKを大特価販売。在庫分のみですので品切れの際には御容赦下さい。

TEL CALL!!

Basic Houseオリジナルハードウェア

for X68000

12bit 8/16ch、高速A/Dコンバータ (Xbasic、XC、アセンブラ用ライブラリ付属)	¥128,000
12bit4-16ch、高速D/Aコンバータ (Xbasic、XC、アセンブラ用ライブラリ付属)	発売予定
16bit絶縁型、パラレルインターフェース (Xbasic、XC、アセンブラ用ライブラリ付属)	¥68,000
64180CPUボード(Mach180) (HD64180/10MHz使用/CP/M80エミュレータ付属)	¥98,000
ハンディプリンタ(Handy PrinJak) (専用インターフェース、ソフト付属)	¥24,800
ユニバーサルボード	¥6,800
ビデオボードケース (CZ-6BV1を外付けにします。)	¥9,800

for X1/turbo

12bit 16ch、高速A/Dコンバータ	¥118,000
12bit 4ch、高速D/Aコンバータ	¥98,000
16bit絶縁型、パラレルインターフェース	¥42,000
GPIBインターフェース	¥58,000
汎用8bit A/D&24pitパラレルI/O	¥19,800
ハードディスクインターフェース	¥16,000

X68000 PRO II-BK

CZ-606Dset
定価¥364,800
特価 ¥2??,000

X68000 PRO -BK

CZ-606Dset
定価¥377,800
特価¥238,000

九十九電機 TS-3XR1

外付け3.5インチフロッピードライブ
定価¥44,800
特価 ¥35,800

EPSON HG-3000

24ドット136桁
高速インクジェットプリンタ
定価¥248,000
特価 ¥98,000

EPSON VP-1350

24ドット136桁
ローコストドットインパクト
定価¥94,000
特価 ¥65,800

CANON BJ-10V

48ドット80桁
ローコストインクジェット
定価¥84,600
特価 ¥69,800

SHARP CZ-8PC5

48ドット80桁カラー熱転写
定価¥96,800
TEL CALL!!

SHARP IO-735X

24ドット136桁
インクジェットカラープリンタ
定価¥248,000
特価¥210,800

SHARP JX-220X

定価¥168,000
特価 ¥143,000

SHARP CZ-8NS1

定価¥188,000
特価 ¥154,000

JX-220X CZ-8NS1 パラレルボード

定価¥29,800
特価 ¥24,800
スキャナとセットならさらに値引き。

ROLAND SC-55/CM-32L

定価¥69,000

ROLAND CM-64

定価¥129,000

TEL CALL!!

SACOM SX-68M

定価¥19,800
特価 ¥16,800

SHARP CZ-6BM1

定価¥26,800
特価 ¥22,800
音源とセットならさらに値引きます。

OMRON MD24FB5V / COMSATRZ CLUB24/5

定価¥39,800
特価 ¥32,800

Basic Houseオリジナルソフトウェア

for X68000

BASIC拡張関数パッケージ (Xbasicの外部関数)	¥9,800
C言語ライブラリ (拡張関数パッケージのC言語版)	¥6,800
BASIC拡張関数パッケージ C言語ライブラリ付き	¥14,800
ディスクキャシャー (SASI HDDとFDDのアクセスを高速化できます。)	¥6,800
CP/M68Kエミュレータ (Human68K上からCP/M68Kのコマンドを実行できます。)	¥19,800

株式会社計測技研/BASIC HOUSE
〒321 栃木県宇都宮市竹林503-1

価格に自信あり!!

OAB特選～X68000シリーズセット　★本体・ディスプレイセットでお買い上げの方にはゲームソフト2本付

①X68000XVI　●SX-WINDOW搭載!!
- CZ-634C-TN
- CZ-614D-TN
- MD-2HD 20枚

定価合計￥503,000
特価 ￥TEL下さい!!

②X68000XVI-HD
- CZ-644C-TN
- CZ-614D-TN
- MD-2HD 20枚

定価合計￥653,000
特価 ￥TEL下さい!!

③X68000 PROII　●SX-WINDOW搭載!!
- CZ-653C-BK/GY
- CZ-606D-BK/GY
- MD-2HD 20枚

定価合計￥364,800
特価￥218,000

④X68000PRO II-HD
- CZ-663C-BK/GY
- CZ-605D-BK/GY
- MD-2HD 20枚

定価合計￥510,000
安くて表示できません。

☆本体、モニターのみの方は、さらにお安くなります。

★X68000XVIお買い上げの方にもれなく数値演算プロセッサー(MB68881RC16)をプレゼントします!!

X68000 SUPER-HD　●SX-WINDOW搭載!!
- SX-WINDOW搭載
- SCSI I/F装備
- 80MBハードディスク搭載
- 3MB大容量メモリ装備
- 高解像度グラフィック

価格応談 表示よりも安く

⑤X68000 SUPER-HD
- CZ-623C-TN(チタン)
- CZ-614D-TN(チタン)
- MD-2HD 20枚

定価合計￥633,000
特価￥366,000

X68000 特選OABセット★本体のみ単品OK!!

価格応談



表示価格よりも安く

① CZ-604C-TN(新品)+CZ-606D-TN(新品)　3セット限り………特価￥268,000
② CZ-604C-TN(新品)+CZ-614D-TN(新品)　1セット限り………特価￥306,000
③ CZ-603C-BK(新品)+CZ-603D-BK(新同品)　3セット限り………特価￥218,000
④ CZ-612C-BK(新品)+CZ-606D-BK(新同品)　2セット限り………特価￥227,000

オーエーブレイン　全国通販

OAB
●オフコンからパソコンまで幅広～い品揃え。おまかせあれ!!　お電話くださいネ!　☎03-5688-3621

★全商品保証書付。専門のアドバイザーがお客様のニーズに親切に対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。
★送料は、着払いとなります。
●ご注文、お問合せは…毎日午前10時から午後8時まで
●下取・買取は電話で見積りしております。責任を持って下取りさせて頂き
●商品のお届けは…入金確認後、即日発送致します。

周辺機器コーナー

プリンターセットコーナー

- CZ-6PVI(カラービデオプリンター) 定価￥198,000……▶特価￥147,800
- CZ-8PC3(24ドット熱転写カラープリンター) 定価￥65,800……▶特価￥52,800
- CZ-8PK10(24ピン漢字ドットプリンター・136桁) 定価￥97,800……▶特価￥70,800
- CZ-8PGI(24ピンカラー漢字ドットプリンター・80桁) 定価￥130,000……▶特価￥91,800
- CZ-8PG2(24ピンカラー漢字ドットプリンター・136桁) 定価￥160,000……▶特価￥113,800
- IO-735XB(カラーイメージェットプリンター) 定価￥248,000……▶特価￥169,000

X68000用ソフトウェアー・コーナー

	定価	特価
①CZ-212BS(BUSINESS)	￥68,000	￥53,000
②CZ-220BS(DATA)	￥58,000	￥45,000
③CZ-215MS(Sampling)	￥17,800	￥13,800
④CZ-221HS(NEW Print Shop)	￥10,800	￥15,500
⑤CZ-227BS(TOP財務会計)	￥200,000	￥158,000
⑥CZ-226BS(CARD)	￥229,800	￥23,000
⑦CZ-223CS(Communicabion)	￥19,800	￥115,500
⑧CZ-213MS(MUSIC)	￥18,800	￥14,800
⑨CZ-211LS(C compiler)	￥39,800	￥31,000
⑩C-TRACE(キャスト)	￥68,000	￥52,000
⑪EW(イースト)	￥38,000	￥29,000

特選品

■CZ-8PC5(48ドット熱転写カラー漢字プリンター)(定価￥96,800)安くて表示できません。

X68000用周辺機器コーナー

型番	品名	(定価)	特価
CZ-6BEI	IBM増設RAMボード	(￥35,000)	￥25,200
CZ-6BEIB	IBM増設RAMボード	(￥28,000)	￥20,200
CZ-6BE2	2MB増設RAMボード	(￥79,800)	￥58,700
CZ-6BE4	4MB増設RAMボード	(￥138,000)	￥102,200
CZ-6BF1	増設用RS-232Cボード	(￥49,800)	￥36,700
CZ-6BGI	GP-IBボード	(￥59,800)	￥43,200
CZ-6BMI	MIDIボード	(￥26,800)	￥19,200
CZ-6BNI	スキャナ用パラレルボード	(￥29,800)	￥21,700
CZ-6BPI	数値演算プロセッサボード	(￥79,800)	￥58,700
CZ-6BOI	ユニバーサルI/Oボード	(￥39,800)	￥29,200
CZ-6EBI/BK	拡張I/Oボックス	(￥88,000)	￥63,700
CZ-6VTI/BK	カラー・イメージ・ユニット	(￥69,800)	￥50,700
CZ-8NM24	マウス	(￥6,800)	￥4,700
CZ-8NTI	マウストラックボール	(￥9,800)	￥6,700
CZ-8NSI	カラーイメージスキャナ	(￥188,000)	￥134,700
CZ-6BCI	FAXボード	(￥79,800)	￥58,700
CZ-8TM2	モデムユニット	(￥49,800)	￥36,700
CZ-64H	増設ハードディスク	(￥120,000)	￥86,700
CZ-6TU GY/BK	RGBシステムチューナー	(￥33,100)	￥23,700
BF-68PRO	高性能CRTフィルタ	(￥19,800)	￥14,700
CZ-6MOI	光磁気ディスクユニット	(￥450,000)	￥326,700
CZ-6BSI	SCSIインターフェースボード	(￥29,800)	￥21,700
CZ-6BL2	LANボード	(￥298,000)	￥216,700

I・O DATA 増設RAMボード　限定

- 1MB増設PAMボード PIO-6BEI-A　定価￥25,000　特価￥16,800
- 2MB増設RAMボード PIO-6BE2-2M　定価￥50,000　特価￥33,300
- 4MB増設RAMボード PIO-6BE4-4M　定価￥88,000　特価￥58,300

ハードディスク　★その他特価品有!TEL下さい!!

- ■シャープ CZ-64H……特価￥86,000
- CZ-68H……特価￥118,000
- ■ロジテック LHD-200……特価￥218,000
- ■アイテム HXD-040……特価￥88,000
- HXD-042……特価￥95,000
- ■アイテック 価格応談 ●TX-80…特価￥77,800
- ●TX-130…特価￥97,800
- ●TX-180…特価￥130,000
- ★SCSIボード………特価￥22,000

オーエーブレイン今月の特価品!! 台数限定 お早目に

- KGB-X68PRKII-02(￥55,000)…特価￥42,800
- PRKII-04(￥90,000)…特価￥70,200
- PRKII-06(￥125,000)…特価￥97,500
- PRKII-08(￥160,000)…特価￥124,800
- PRKII-12(￥85,000)…特価￥66,300
- PRKII-14(￥120,000)…特価￥93,600
- PRKII-16(￥155,000)…特価￥121,000
- PRKII-18(￥190,000)…特価￥148,000
- MC-6888 IRC (￥38,000)…特価￥28,500

〈開発ツール〉●C-コンパイラPRO68KV.2 定価￥44,800 CZ-245IS ……特価￥33,000
〈C言語〉●C&Professional Pack 定価￥58,000 ……特価￥40,500
〈データベース〉●CARD PRO68K Ver.2.0 定価￥29,800 CZ-253BS ……特価￥21,000
〈音楽〉●Music studio PRO68K Ver.2.0 定価￥28,800 CZ-261MS ……特価￥21,300
〈CGシール〉●CANVAS PRO68K 定価￥29,800 CZ-249GS ……特価￥22,200
〈通信〉●Tlepotion PRO68K 定価￥22.800 CZ-258BS ……特価￥
〈ワープロ〉●Multiword PRO68K 定価￥32,000 CZ-225BS ……特価￥
〈グラフィック〉●Z's STAFF PRO68 (シャフト)定価￥58,000 ……特価￥
〈グラフィック〉●C-TRACE68 Ve 定価￥98,000 ……特価￥

通信販売によるご購入方法(お電話でお申し込み下さい。)

現金一括払い
銀行振込：電信扱いにてお振込下さい　手数料はお客様負担となります
現金書留：住所、氏名、電話番号、商品名、使用機種、メディア等をお書き添えのうえ、現金書留にて当社までお送り下さい

クレジット
専用のお申し込み用紙をお送り致しますので、必要事項をご記入・捺印のうえ、ご返送下さい
※未成年者の方は、保護者のご承認を受けてからお申し込み下さい

振込先
●第一勧業銀行 御徒町支店 (普)1376679 オーエーブレイン
●朝日信用金庫 本店 (普)334833 オーエーブレイン

★クレジットは1～60回払いで月々5,000円よりご自由に設定できます。

OAB オーエーブレイン　〒110 東京都台東区台東1-28-4　TEL & FAX 5688-3621

中古パソコン

PC-9801RA2	￥198,000より
PC-9801RX2	￥130,000より
PC-9801VX21	￥125,000より
PC-9801VM21	￥90,000より
PC-9801VM2	￥75,000より
PC-9801F2	￥18,000より
PC-9801EX2	￥130,000より
PC-9801UV21	￥65,000より
PC-9801LV21	￥93,000より
PC-286V	￥75,000より
PC-286VE	￥80,000より
PC-286L	￥80,000より
PC-286LS	￥190,000より
PC-8801FH	￥18,000より
PC-8801MA	￥25,000より
X68000	￥110,000より
X68000(HD)	￥160,000より
X1ターボZII	￥28,000より
FM77AV40EX	￥15,000より
200ラインCRT	￥8,000より
400ラインCRT	￥30,000より
80桁プリンタ	￥15,000より
135桁プリンタ	￥35,000より

●その他多数有り、お問い合せ下さい。

ユニット

- FD-1155D(5インチ)……￥9
- FD-1155C(5インチ)……￥8
- FD-1165A(8インチ)……￥3
- FD-1137D(3.5インチ)……￥9
- D-5146H(5インチ40MB)…￥29
- D-3142(3.5インチ40MB)…￥29
- D-3148(3.5インチSISC)……￥30
- 外付8インチ2ドライブ……￥20
- 外付5インチ2ドライブ……￥30

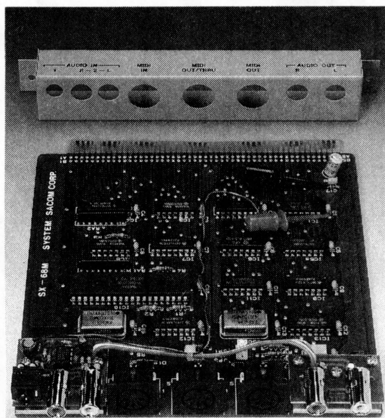
■流通事情により、広告表示よりお安くなる場合もございます。まずは、お電話下さい。■ビジネス・ゲームセットもございます。

良い音追求！ SX-68MIX

通信販売のみ！

"オーディオミキサー"内蔵MIDIボード(聴くだけソフト付)登場！

¥23,690(送料、税込)



MIDI対応ゲームのSE(効果音)はADPCMやFM音源を使用しています。MIDI音源を増やすと接続機器が多くなり接続の矛盾が生じます。SX-68 MIXの専用ミキサーを使用すればX68000のLINE OUTとMIDI音源のOUTをミックスし、素晴らしいオーディオ環境をつくります。

〔仕様〕

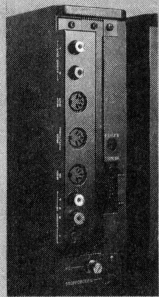
■オーディオミキサー／入力:MIDI音源用STEREO AUDIO IN×1(RCAピンプラグ)、内蔵音源用STEREO AUDIO IN×1(ミニステレオプラグ) 出力:STEREO AUDIO OUT×1(RCAピンプラグ) ■オリジナルスロットカバー(グレーまたはブラック) ■ステレオコード(1mミニステレオプラグ) ■SX-68M(システムサコム社製)X68000用MIDIボード ■聴くだけソフト／Mu-1、Musicstudio PRO-68K(SNGファイル)、MUSIC PRO-68K(MUSファイル)、MML(OPMDファイル)、ミュージ郎(98用SNGファイル)、スタンダードMIDIファイル(MIDファイル)以上5種類のデータを読み込み再生することができます。

オーディオミキサー拡張キット

¥8,240(送料、税込)

すでにSX-68M(MIDIボード)をお持ちの方の為に"オーディオミキサー"拡張キットも用意しました。

※注意:CZ-6BM1(シャープ社製MIDIボード)に取り付けることはできません。



〔仕様〕

■オーディオミキサー
■オリジナルスロットカバー(グレーまたはブラック)
■ステレオコード(1mミニステレオプラグ)

販売方法は通信販売のみです。お申し込みは郵便振替をご利用ください。

●郵便振替 口座番号"東京1-651415 株式会社サンミュージカルサービス"
注意:払込手数料はお客様がご負担してください。(青色の払込通知書を使用)

●必要事項の記入
住所、氏名、電話番号(自宅及び会社)、払込金額、スロットカバーの色(グレーまたはブラック)をご記入してください。



SAN MUSICAL SERVICE
〒154 東京都世田谷区池尻3-21-28池尻成和ビル
TEL03-3419-8839

X68000 MAC 開発スタッフ募集！

X68000がしゃべります　¥2,000
SPEAK SYSTEM

◆かな文字を再生する SPK デバイスドライバー
かな文字のファイルをコピーすると読み上げます。

◆メモリ上のデータを再生する PRG ドライバー
指定された区間を連続して再生します。

◆PCM と BAT ユーティリティ１６本付属
SPEAK ユーティリティを使用して
SPEAK おはよう
と入力すれば「おはよう」としゃべります。

◆レクチャー付属
使用法を実際に操作しながら簡単に学べます。

◆DiSS-Pのデータも再生も再生可能

今回、案内状の届かなかったDiSS-Pのユーザーは現在の住所をご連絡下さい。

X68000のコマンドライン言語　¥2,000
FANCTION CALL

◆アセンブラやコンパイラ不要
コマンドラインから入力するだけで IOCS・DOS・FE の各ファンクションコールを簡単に呼び出せます。

◆テキスト画面に四角形を描く書式例
CALL TXBOX A1=0.w, 0.w, 0.w, 80.w, 80.w, $FFFF.w;

◆戻り値はバッチファイルで使用可能
D0 のワードデータが参照できます。バッチファイルに組み込めば簡単なプログラムの製作も可能です。

◆レクチャー付属
使用法やファンクションコールに必要な知識を実際に操作しながら簡単に学べます。

◆豊富な実用例
レクチャー全体がバッチファイルで書かれているので誰でも参照し、利用できます。

通信販売

商品名を明記して郵便振替又は現金書留でご注文ください。(税込・送料サービス)
(郵便振替 東京 8-404042)

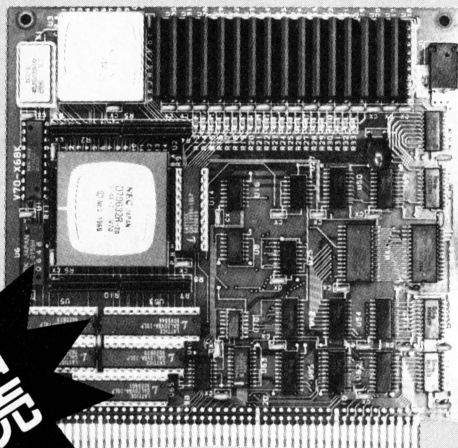
サザン エンタープライズ
〒142 東京都品川区戸越5-12-17 TEL・FAX 03-3787-3932

ACCESS

あなたのX68000が

Superワークステーションに!

V70 +AFPP アクセラレータ



近日発売

ボードスペック

- V70 CPU 20MHz(μPD70632)
- V70 AFPP(μPD72691)
  フローティング・ポイント・プロセッサ
- DRAM 2MByte
  同一ページ内アクセスはNo Wait
- SRAM 128KByte
  X68000と共有

同梱ソフト

- アセンブラ
- モニタ
- ソースコードデバッガ
- フロートエミュレータ

オプションソフト

- Cコンパイラ

AFPP(フローティング・ポイント・プロセッサ)標準搭載

最速10MIPS※1 5.8M FLOPS※2

無限の可能性を秘め

この冬X68000上に堂々デビュー

V70のあらたなるパワーをあなたの手に!



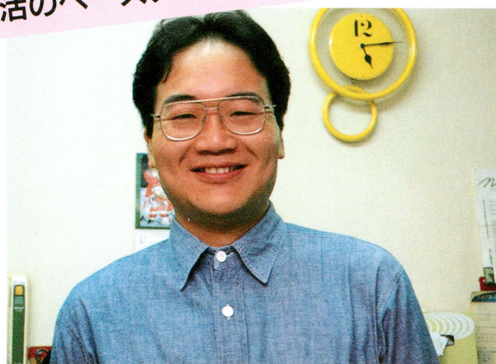
※1 V70 : レジスターレジスタ間基本命令、NOP命令(実測)
※2 AFPP : ベクトル/行列演算(倍精度)、μPD72691ユーザーズ・マニュアルより
●本製品は、有限会社アクセスと株式会社ハドソンの共同開発製品です。

有限会社アクセス 〒101 東京都千代田区神田神保町1-64
神保町協和ビル7F
☎03(3233)0200(代) FAX.03(3291)7019

こ・ん・に・ち・は ネットワーカー⑥

パソコン/ワープロ通信ネットワークサービス J&P HOTLINE

J&P HOTLINEは
生活のペースメーカー。



片山　拓也さん　30歳

(JH001368 パタリロ)テクニカルライター

J&P HOTLINEのSIGOPをされている片山さん。テクニカルライターとして精力的に仕事をこなされると同時に、機械・法律を扱うライターとして、そして自ら主催する草の根ネットワークのSYSOPとして八面六臂の活躍です。さすがはネットワーカーとも言える片山さんの通信生活をのぞいてみました。

公私ともどもネットワーカー。通信が産んだ生え抜きライター。

片山さんは、毎日2回必ず通信されるとか。J&P HOTLINEはもちろんのこと、他のネットや仕事で取引先と活用しているクローズドなネットワークまで、自分あての電子メールを確認するためだそうです。まさに片山さんにとって通信は、なくてはならない仕事の道具といえる存在です。

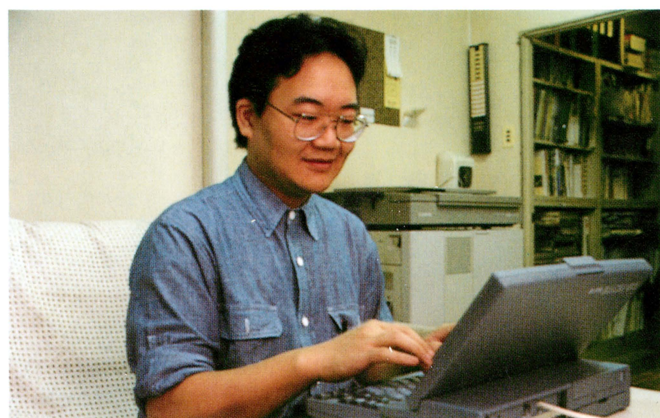
「通信の上でいろいろな方と知り合いになれて、それが仕事にも役だっている。それがなによりの通信のメリットですね」と語る片山さん。通信の上での意見が目にとまり、仕事に結びついたこともおありだとか。大学を出てすぐに大手電機メーカーのマニュアルを担当。それ以来コンピュータとは切っても切れぬ仲。HOTLINEでさまざまなテクニカル情報を手にいれられることもあるという片山さんは、まさに通信とともに育ってきた、日本のテクニカルライターの創生期からの生え抜きライターともいえる存在です。

「通信」へのかかわりは、いまではSTAR FAXというパソコン用FAXボードのマニュアル・パッケージなどの仕事に結実。

奥様とも、HOTLINEの電子メールで知りあわれたといいますから、まさに仕事とプライベートの両面でHOTLINEは片山さんの生活の中心です。

もともとはハンダごて少年だったという片山さん。その経験が活きてか、今ではみずから草の根ネットも開設。とくにこのネットでは、クローズドなライターだけのボードを設定。仕事上での情報交換に活用されているとか。このネットは今や発展成長し、商業ネットとして再出発するという計画まであるそうです。

「最近やっと通信の上で素直に自己表現できるようになりましたからね」とおっしゃる片山さん。言葉だけでコミュニケートするパソコン通信は、片山さんにとってもっとも自分を出せるメディアなのかもしれません。



マルチな片山さんには、HOTLINEの多様なサービスがジャストフィット。

★日本全国の情報もネットワーキングで手に入れる。地域ボードも片山さんのお気に入り。

★SIG間の交流にも熱心な片山さんのまわりには素敵なネットワーカーがいっぱい。コミュニケーションはすべての基本です。

★仕事のツールはネットで探す。XMODEMがあるのでフリーソフトも手に入ります。

J&P HOTLINEへのご入会はスタータキットで。

買ったその日から　2週間無料で　アクセスできます。

お求めは、下記のお店へ。又は現金書留にて、¥3,000+¥90(消費税3%)=¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは　〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社
J&P HOTLINE事務局宛　TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

渋谷店	東京都渋谷区道玄坂2丁目28番4号	☎(03)3496-4141
町田店	東京都町田市森野1丁目39番16号	☎(0427)23-1313
八王子店	東京都八王子市旭町1番1号八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4－39－1	☎(0425)36-4141
本厚木店	厚木市中町3－4－3	☎(0462)25-1548
富山店	富山市掛尾町300番地	☎(0764)22-5033
金沢店	金沢市入江2－63	☎(0762)91-1130
寺地店	金沢市寺地2－3	☎(0762)47-2524
大須店	名古屋市中区大須4丁目2-48	☎(052)262-1141
テクノランド	大阪市浪速区日本橋5丁目6番7号	☎(06) 634-1211
メディアランド	大阪市浪速区日本橋5丁目8番26号	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2丁目1番17号	☎(06) 634-3111
U. S. LAND	大阪市浪速区日本橋4丁目9番15号	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
梅田店	大阪市北区小松原町1－10	☎(06) 362-1141
高槻店	高槻市高槻町11番16号	☎(0726)85-1212
くずは店	枚方市楠葉花園町15番2号	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3 SENCHU PAL 2番街4F	☎(06) 834-4141
摂津富田店	高槻市大畑町24－10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4－20	☎(0720)34-1166
藤井寺店	藤井寺市岡2丁目1番33号	☎(0729)38-2111
岸和田店	岸和田市土生町2451－3	☎(0724)37-1021
さんのみや1ばん館	神戸市中央区八幡通3-2-16	☎(078)231-2111
西宮店	兵庫県西宮市河原町5－11	☎(0798)71-1171
伊丹店	伊丹市昆陽池1－63	☎(0727)77-5101
姫路店	姫路市東延末1丁目1番住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵比須之町549	☎(075)341-3571
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4丁目4番地	☎(0734)28-1441
奈良1ばん館	奈良市三条町478－1	☎(0742)27-1111
郡山インター店	大和郡山市横田693－1	☎(07435)9-2221
熊本店	熊本市手取本町4－12	☎(096)359-7800

T4910217912606 雑誌 02179-12